HashiCorp Terraform Associate (003) Exam Guide

Preface:

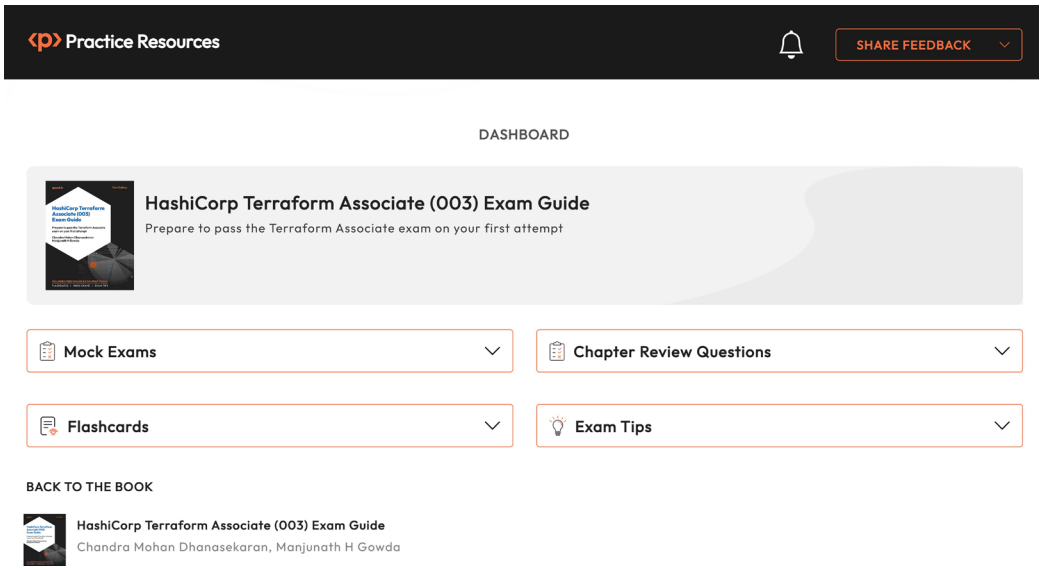

Figure 0.1 – Online exam-prep platform on a desktop device

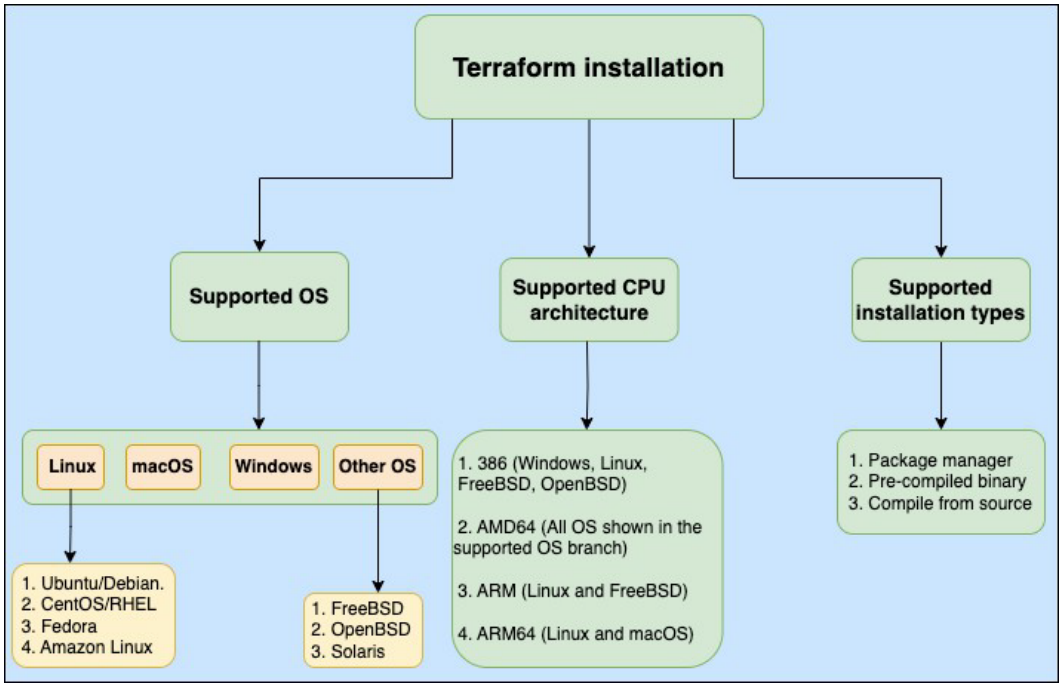

Figure 0.2: Terraform installation

```
~$brew install hashicorp/tap/terraform
==> Fetching hashicorp/tap/terraform
==> Downloading https://releases.hashicorp.com/terraform/1.7.3/terraform_1.7.3_darwin_arm64.zip
Already downloaded: /Users/manjunath/Library/Caches/Homebrew/downloads/a0ca203fddc90ae140c9d76ebdf8991f960ab
3c01a8baf1df7626550e389b7d6--terraform_1.7.3_darwin_arm64.zip
==> Installing terraform from hashicorp/tap
==> Downloading https://formulae.brew.sh/api/formula.jws.json
######################################################################################################## 100.0%
==> Downloading https://formulae.brew.sh/api/cask.jws.json
######################################################################################################## 100.0%
Warning: Cask homebrew/cask/terraform was renamed to homebrew/core/terraform.
🍺  /opt/homebrew/Cellar/terraform/1.7.3: 3 files, 88.6MB, built in 4 seconds
==> Running `brew cleanup terraform`...
Disable this behaviour by setting HOMEBREW_NO_INSTALL_CLEANUP.
Hide these hints with HOMEBREW_NO_ENV_HINTS (see `man brew`).
~$
```

Figure 0.3: Terraform installation using Homebrew

s

```
~$terraform -v
Terraform v1.7.3
on darwin_arm64
+ provider registry.terraform.io/hashicorp/aws v4.64.0
~$_
```

Figure 0.4: Terraform installation validation

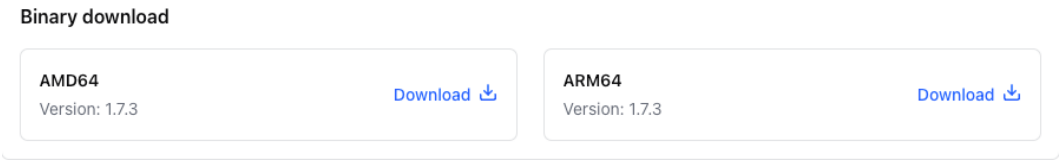

Figure 0.5: Pre-compiled binary for macOS

```
~/Downloads $ls
terraform_1.7.3_darwin_arm64.zip
~/Downloads $
~/Downloads $unzip terraform_1.7.3_darwin_arm64.zip
Archive:  terraform_1.7.3_darwin_arm64.zip
  inflating: terraform
~/Downloads $
~/Downloads $ls --color=auto
terraform                      terraform_1.7.3_darwin_arm64.zip
~/Downloads $
```

Figure 0.6: Unzipping the Terraform ZIP file

```
~/Downloads $terraform -v
Terraform v1.7.3
on darwin_arm64
```

Figure 0.7: Pre-compiled binary installation validation

```
PS C:\Users\Administrator> choco install terraform
Chocolatey v2.2.2
Installing the following packages:
terraform
By installing, you accept licenses for the packages.
Progress: Downloading terraform 1.7.3... 100%

terraform v1.7.3 [Approved]
terraform package files install completed. Performing other installation steps.
The package terraform wants to run 'chocolateyInstall.ps1'.
Note: If you don't run this script, the installation will fail.
Note: To confirm automatically next time, use '-y' or consider:
choco feature enable -n allowGlobalConfirmation
Do you want to run the script?([Y]es/[A]ll - yes to all/[N]o/[P]rint): Y

Removing old terraform plugins
Downloading terraform 64 bit
  from 'https://releases.hashicorp.com/terraform/1.7.3/terraform_1.7.3_windows_amd64.zip'
Progress: 100% - Completed download of C:\Users\Administrator\AppData\Local\Temp\chocolatey\terraform\1.7.3\terraform_1.
7.3_windows_amd64.zip (25.05 MB).
Download of terraform_1.7.3_windows_amd64.zip (25.05 MB) completed.
Hashes match.
Extracting C:\Users\Administrator\AppData\Local\Temp\chocolatey\terraform\1.7.3\terraform_1.7.3_windows_amd64.zip to C:\
ProgramData\chocolatey\lib\terraform\tools...
C:\ProgramData\chocolatey\lib\terraform\tools
 ShimGen has successfully created a shim for terraform.exe
```

Figure 0.9: Terraform installation via Chocolatey

1.

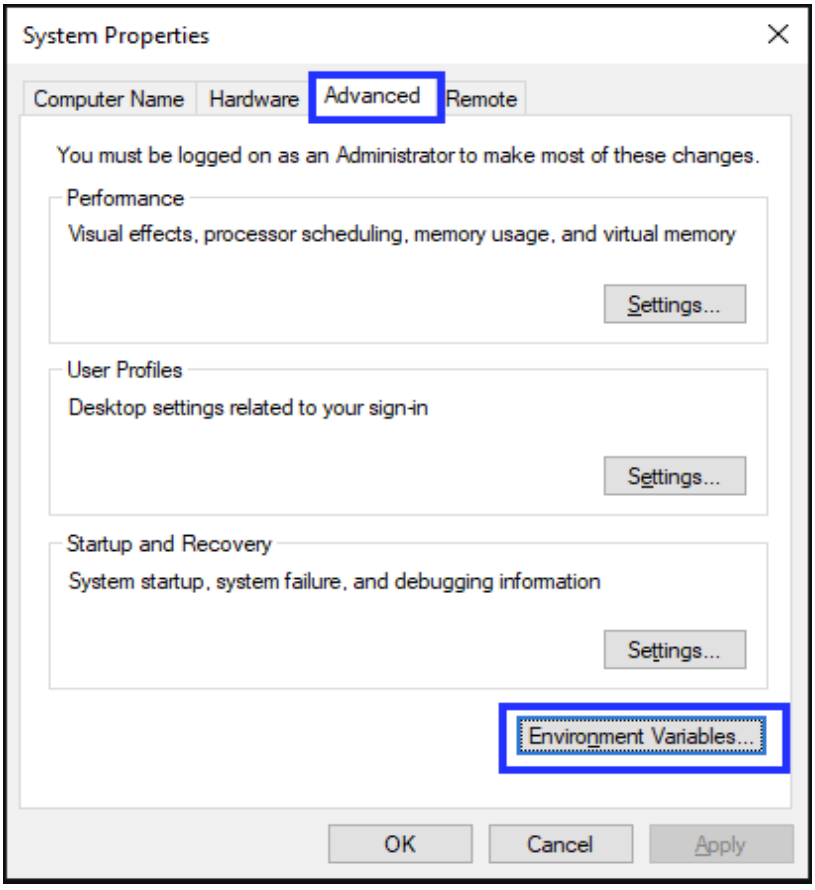

Figure 0.10: System properties

1.

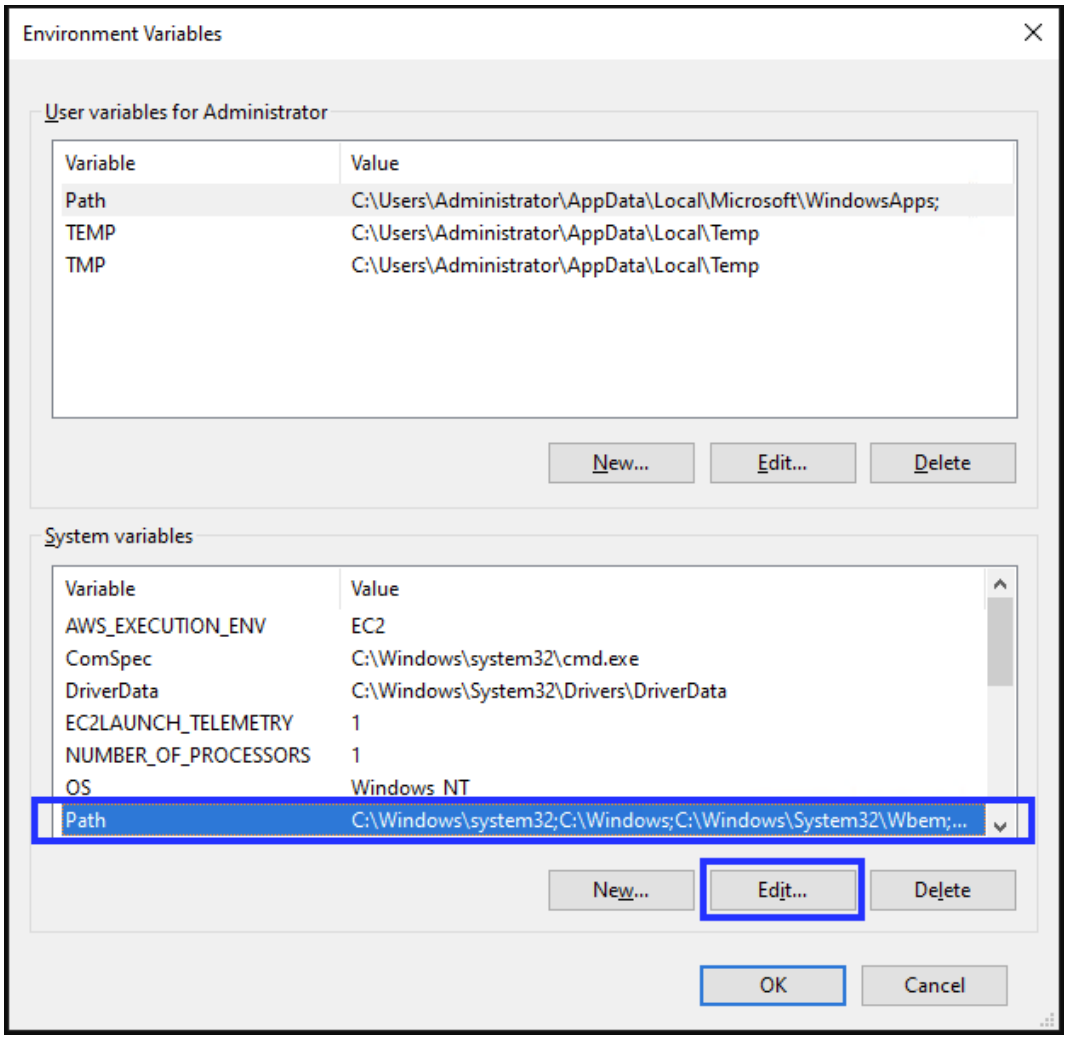

Figure 0.11: Modifying the PATH environment variable

2.

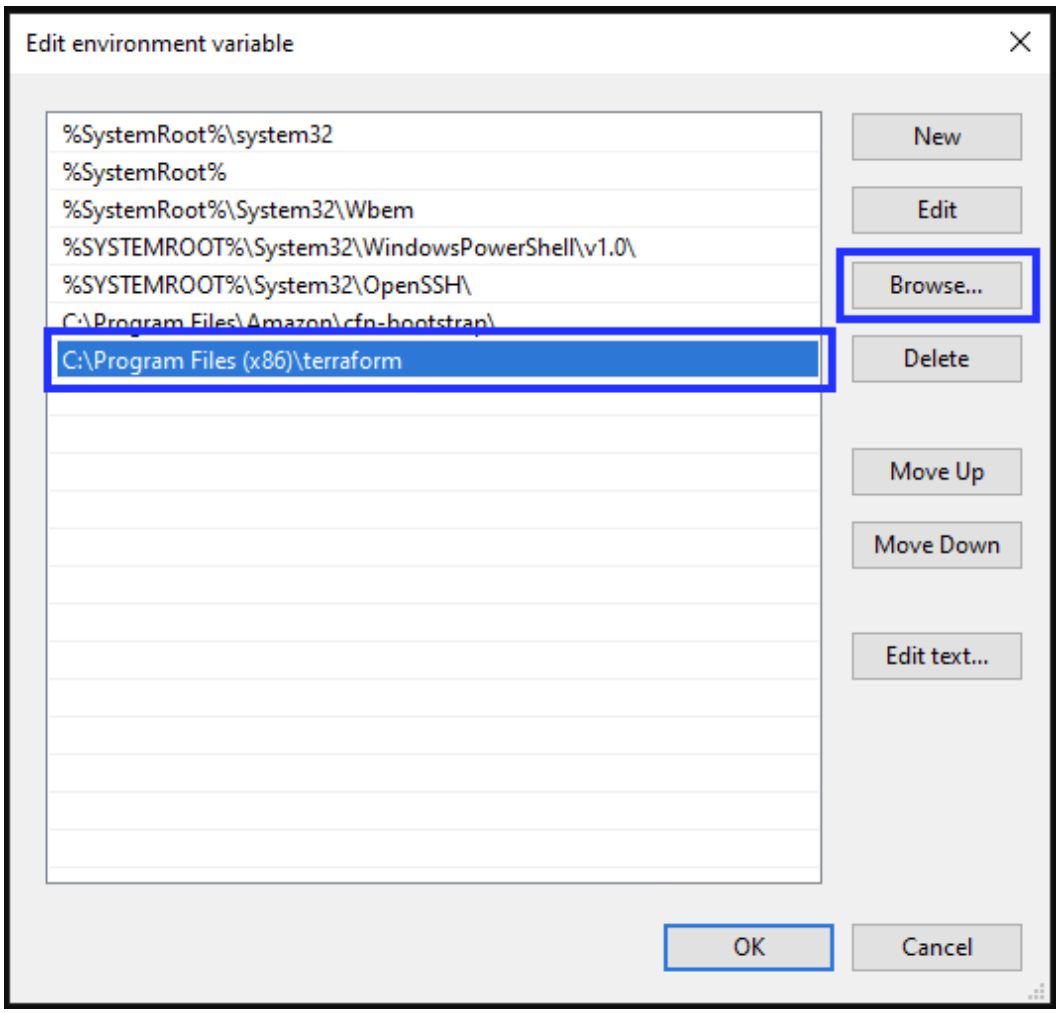

Figure 0.12: Adding the Terraform location to the PATH environment variable

```
~ $aws configure
AWS Access Key ID [None]: AKIAYC3[redacted]
AWS Secret Access Key [None]: EdMKRvx4OS5[redacted]4VJo5NzUO5gA
Default region name [None]: ap-south-1
Default output format [None]: json
~ $
```

Figure 0.13: Configuring the AWS CLI

```
/Users/packt/Desktop $terraform init

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/aws versions matching "~> 5.0"...
- Installing hashicorp/aws v5.37.0...
- Installed hashicorp/aws v5.37.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
/Users/packt/Desktop $terraform plan

No changes. Your infrastructure matches the configuration.
```

Figure 0.14: Configuring AWS provider

```
  # module.vpc.aws_vpc.this[0] will be created
  + resource "aws_vpc" "this" {
      + arn                                  = (known after apply)
      + cidr_block                           = "10.0.0.0/16"
      + default_network_acl_id               = (known after apply)
      + default_route_table_id               = (known after apply)
      + default_security_group_id            = (known after apply)
      + dhcp_options_id                      = (known after apply)
      + enable_dns_hostnames                 = true
      + enable_dns_support                   = true
      + enable_network_address_usage_metrics = (known after apply)
      + id                                   = (known after apply)
      + instance_tenancy                     = "default"
      + ipv6_association_id                  = (known after apply)
      + ipv6_cidr_block                      = (known after apply)
      + ipv6_cidr_block_network_border_group = (known after apply)
      + main_route_table_id                  = (known after apply)
      + owner_id                             = (known after apply)
      + tags                                 = {
          + "Name" = ""
        }
      + tags_all                             = (known after apply)
    }

Plan: 4 to add, 0 to change, 0 to destroy.
```

Figure 0.15: Terraform plan for VPC creation in AWS

Chapter 1: Introduction to Infrastructure as Code (IaC) and Concepts

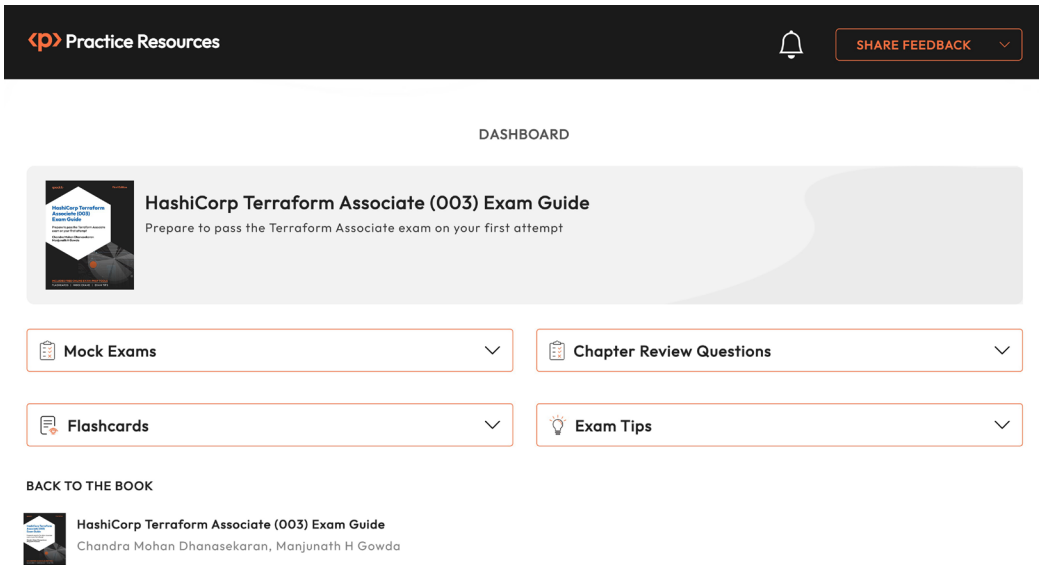

Figure 1.1 – Dashboard interface

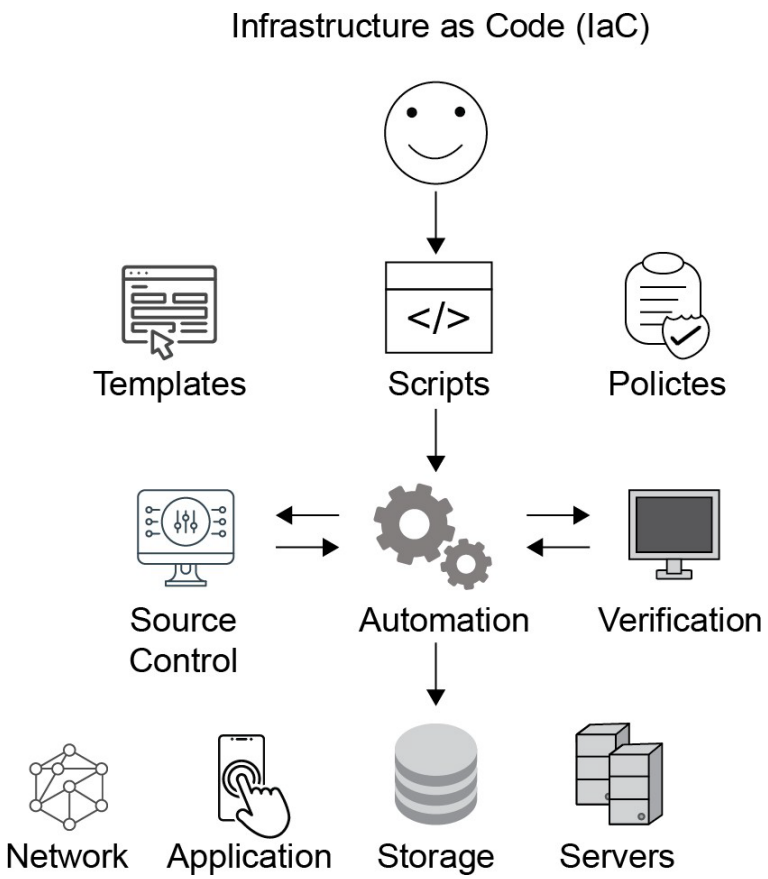

Figure 1.2 – IaC workflow

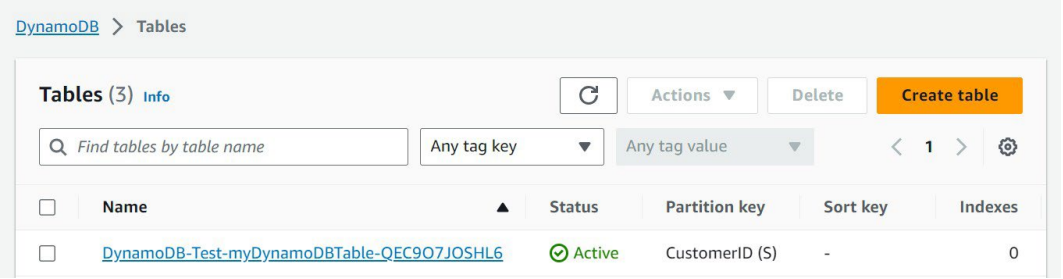

Figure 1.3 – Table in the console

```
Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/aws versions matching "~> 5.14"...
- Installing hashicorp/aws v5.15.0...
- Installed hashicorp/aws v5.15.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

Figure 1.4: The output will look like this

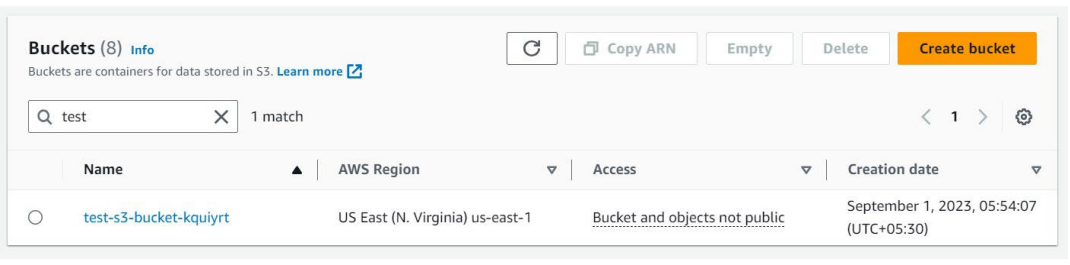

Figure 1.5 – Bucket in the console

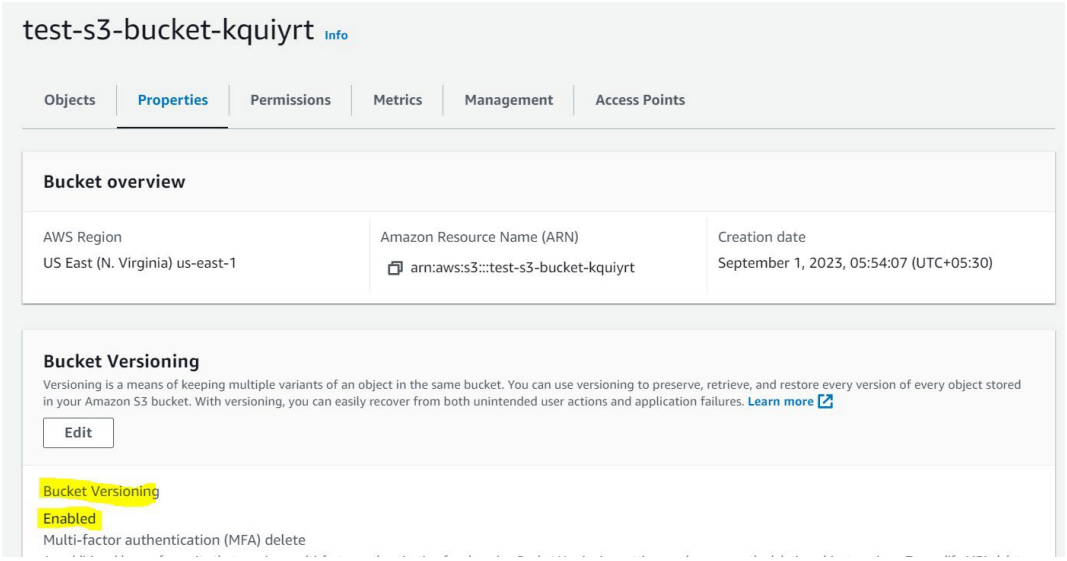

Figure 1.6 – S3 bucket with versioning option enabled

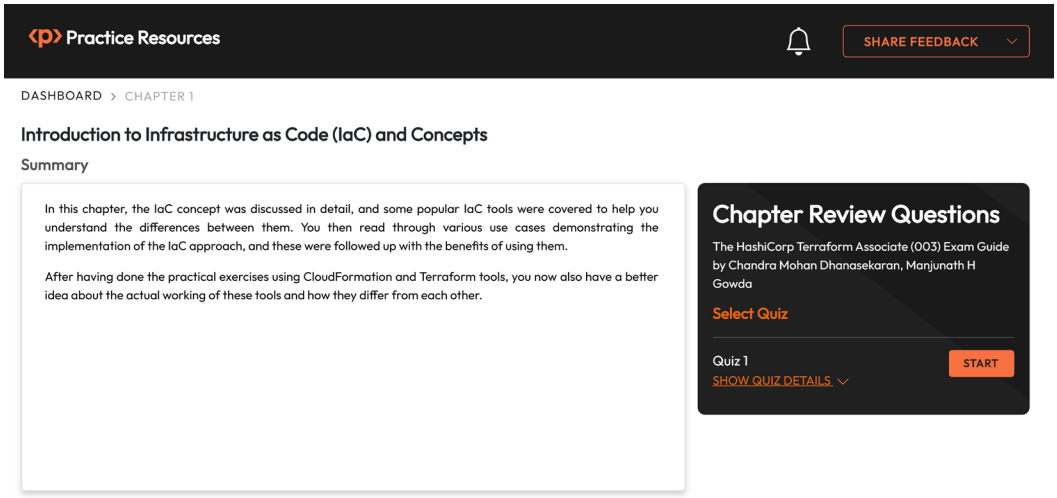

Figure 1.8 – Chapter Review Questions for Chapter 1

Chapter 2: Why Do We Need Terraform?

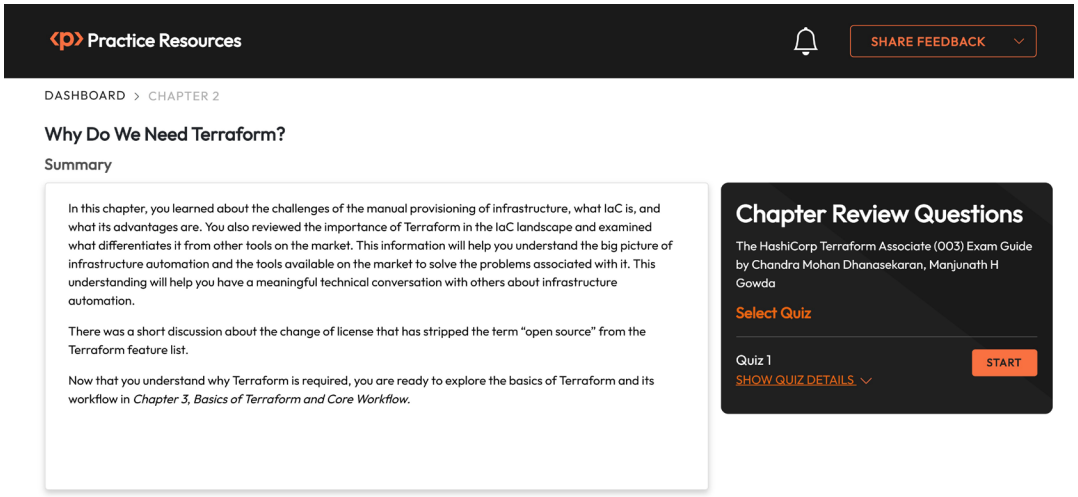

Figure 2.2 – Chapter Review Questions for Chapter 2

Chapter 3: Basics of Terraform and Core Workflow

Figure 3.1 – A high-level overview of Terraform providers

```
C:\Users>terraform -version
Terraform v1.5.7
on windows_amd64
```

Figure 3.2 – Terraform version command output

```
C:\Users>aws --version
aws-cli/2.15.16 Python/3.11.6 Windows/10 exe/AMD64 prompt/off
```

Figure 3.3 – AWS CLI version command output

```
Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.19.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

Figure 3.4 – terraform init command output

```
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
  + create

Terraform will perform the following actions:

  # aws_iam_access_key.user_access_keys will be created
  + resource "aws_iam_access_key" "user_access_keys" {
      + create_date                    = (known after apply)
      + encrypted_secret               = (known after apply)
      + encrypted_ses_smtp_password_v4 = (known after apply)
      + id                             = (known after apply)
      + key_fingerprint                = (known after apply)
      + secret                         = (sensitive value)
      + ses_smtp_password_v4           = (sensitive value)
      + status                         = "Active"
      + user                           = "test-aws-user"
    }
```

```
  # aws_iam_user.test_user will be created
  + resource "aws_iam_user" "test_user" {
      + arn           = (known after apply)
      + force_destroy = false
      + id            = (known after apply)
      + name          = "test-aws-user"
      + path          = "/test/"
      + tags          = {
          + "createdby" = "terraform"
        }
      + tags_all      = {
          + "createdby" = "terraform"
        }
      + unique_id     = (known after apply)
    }

Plan: 2 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value:
```

Figure 3.5 – terraform apply console output

```
aws_iam_user.test_user: Creating...
aws_iam_user.test_user: Creation complete after 1s [id=test-aws-user]
aws_iam_access_key.user_access_keys: Creating...
aws_iam_access_key.user_access_keys: Creation complete after 0s [id=                    ]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
```

Figure 3.6 – resource creation from terraform apply

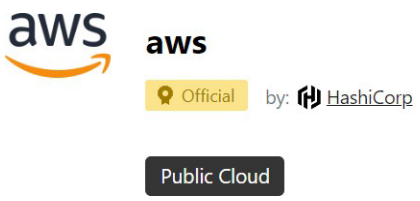

Figure 3.7 – Official provider: aws

Figure 3.8 – Partner provider: oci

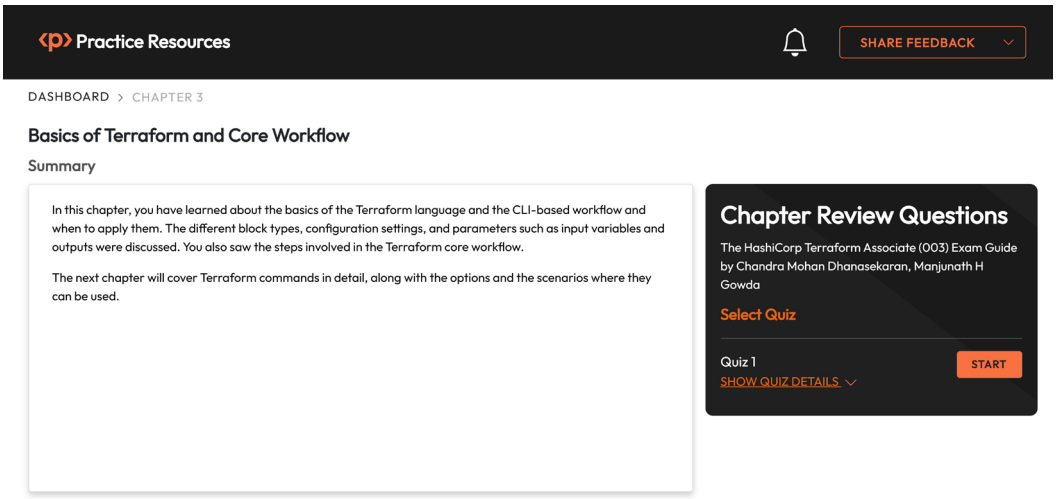

Figure 3.11 – Chapter Review Questions for Chapter 3

Chapter 4: Terraform Commands and State Management

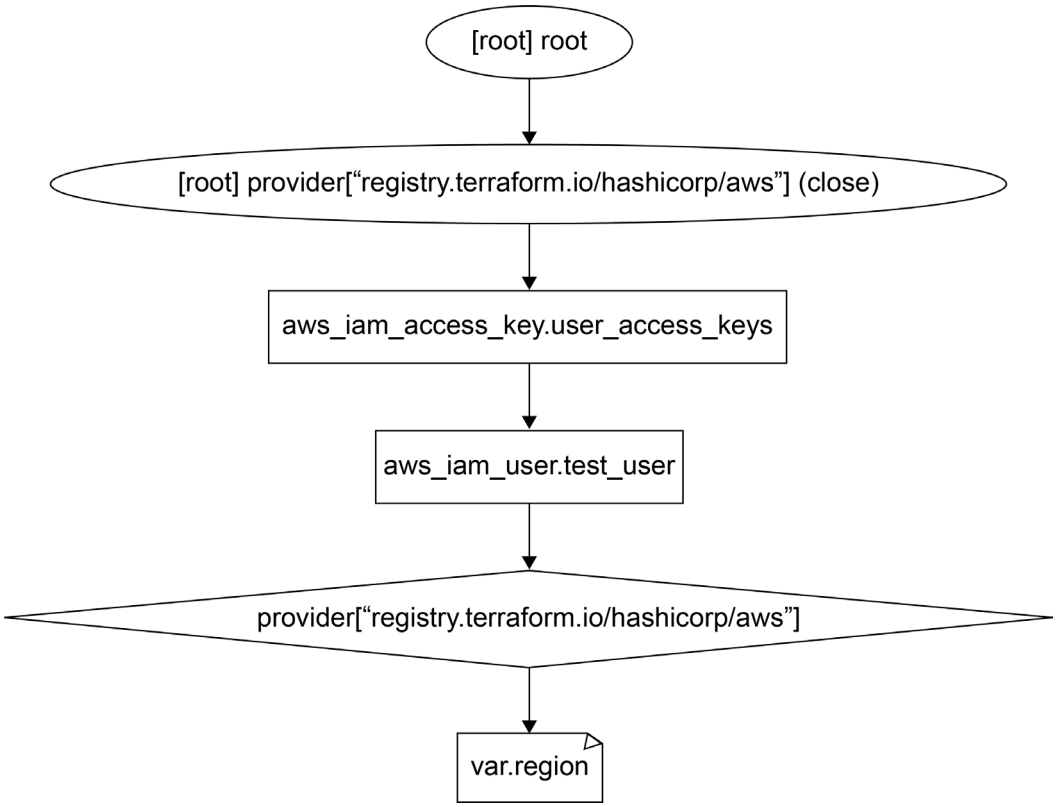

Figure 4.1: The graph command output

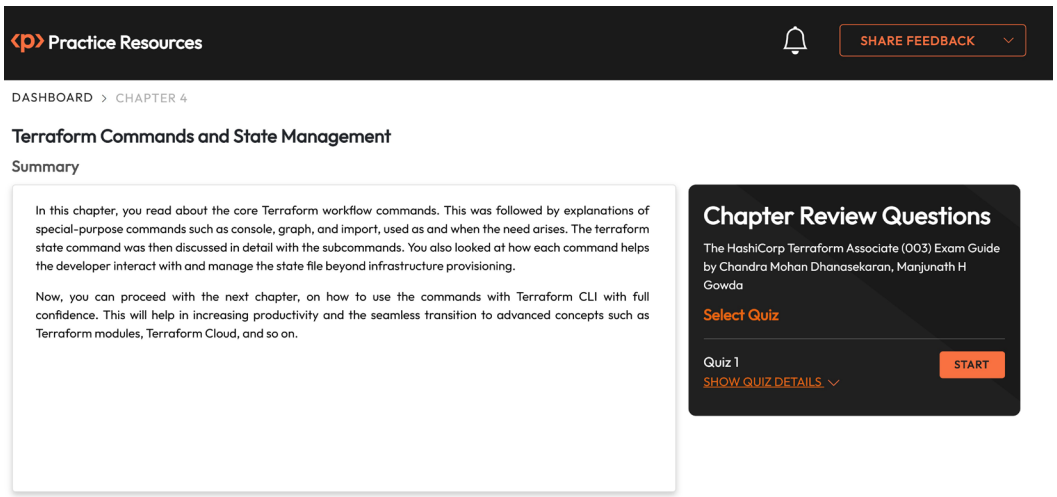

Figure 4.3 – Chapter Review Questions for Chapter 4

Chapter 5: Terraform Modules

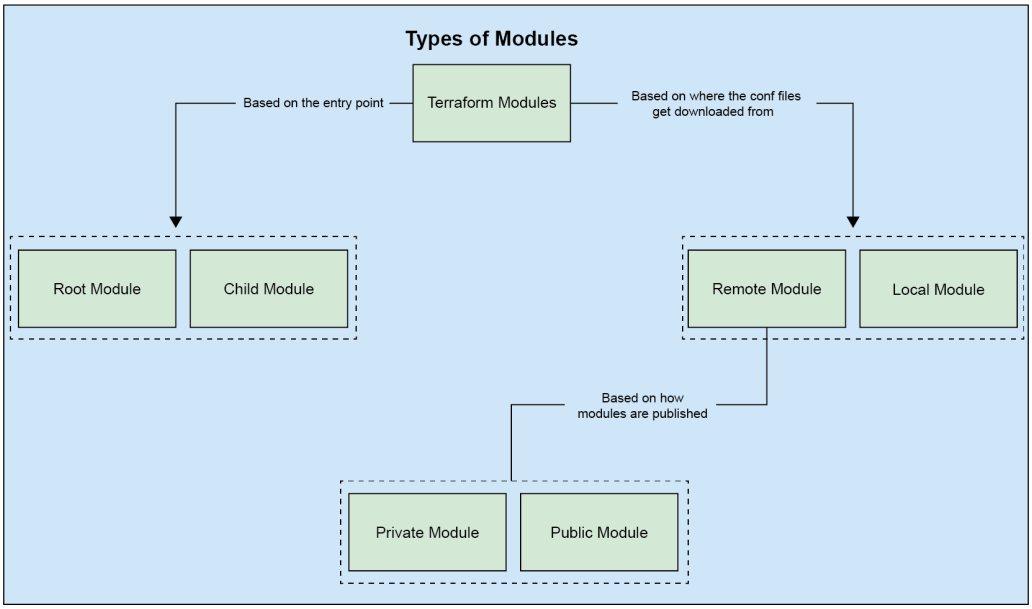

Figure 5.1: Types of modules

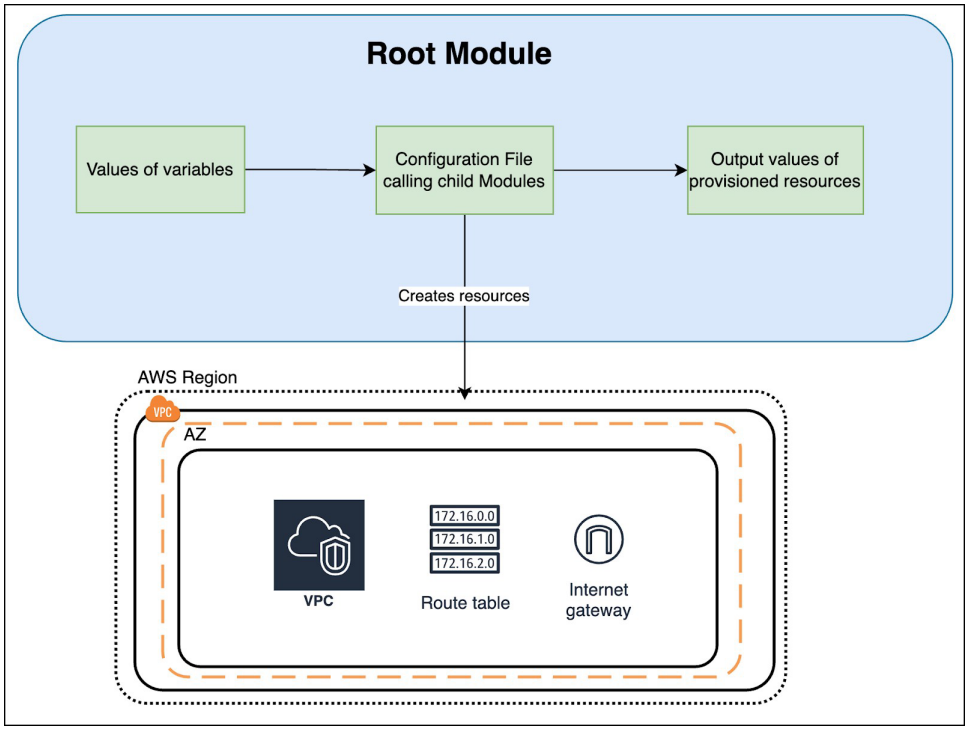

Figure 5.2: Root module

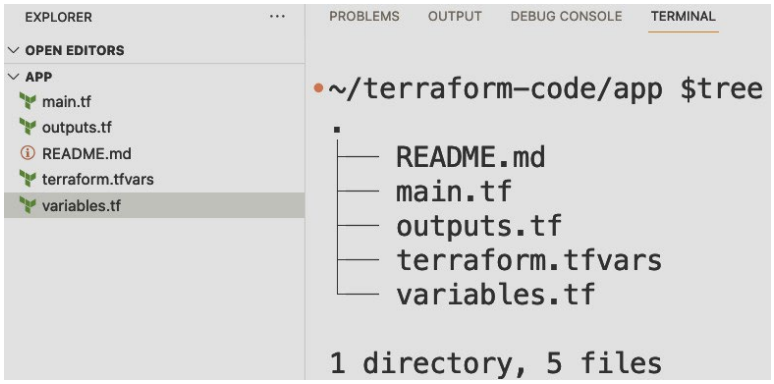

Figure 5.3: Root module file structure example

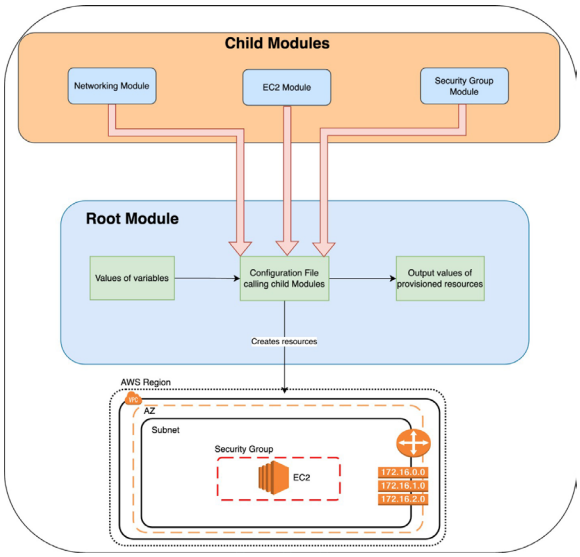

Figure 5.4: Child module called by root module

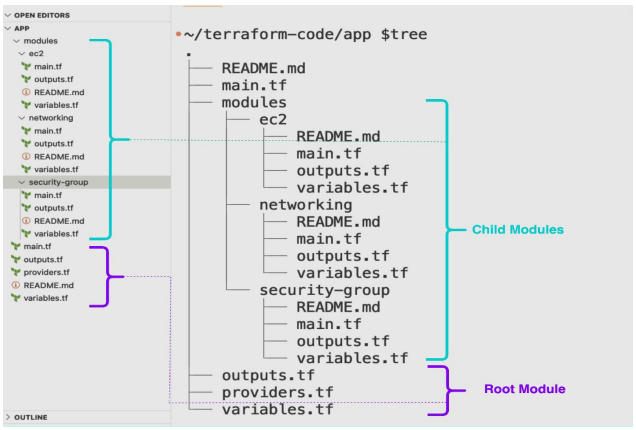

Figure 5.5: Child module and root module file structure example

```
###  CREATING VPC, SUBNET, ROUTE TABLE, IGW ###
module "vpc" {
  source            = "./modules/networking"
  vpc_cidr_block    = "10.51.0.0/16"
  vpc_name          = "child-module-vpc"
  subnet_cidr_block = "10.51.1.0/24"
  az                = "ap-south-1a"
  igw_name          = "child-module-igw"
}

###  CREATING SECURITY GROUP AND RULES ###
module "security_group" {
  source              = "./modules/security-group"
  vpc_id              = module.vpc.vpc_id
  security_group_name = "child-module-sg"
  allowed_ip          = "0.0.0.0/0"
  from_port           = "22"
  to_port             = "22"
}
```

source path is within the local filesystem

Figure 5.6: Example code for local module implementation

```
~/terraform-code/app $terraform init

Initializing the backend...
Initializing modules...
- ec2_instance in modules/ec2
- security_group in modules/security-group
- vpc in modules/networking

Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.34.0"...
- Installing hashicorp/aws v5.34.0...
- Installed hashicorp/aws v5.34.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!
```

Local child modules getting initialized

Figure 5.7: Local child modules getting initialized

```
module "ec2_instance" {
  source                 = "terraform-aws-modules/ec2-instance/aws"
  version                = "5.6.0"
  name                   = "public-module-ec2"
  ami                    = data.aws_ami.root_module_ami.id
  instance_type          = "t3.micro"
  subnet_id              = module.vpc.public_subnets[0]
  vpc_security_group_ids = [module.security_group.security_group_id]
  key_name               = aws_key_pair.root_module_kp.id
  tags = {
    Name = "public-module-ec2"
  }
}
```

source path is referring to the implied terraform public registry for child module

Figure 5.8: Example code for a public (and remote) module (Terraform Registry)

main.tf

```
1  module "example" {
2      source = "github.com/manju712/terraform-remote-module"
3      }
4
```

Source path is referring to the GitHub's repository.

Figure 5.9: Example code for public (and remote) module (GitHub)

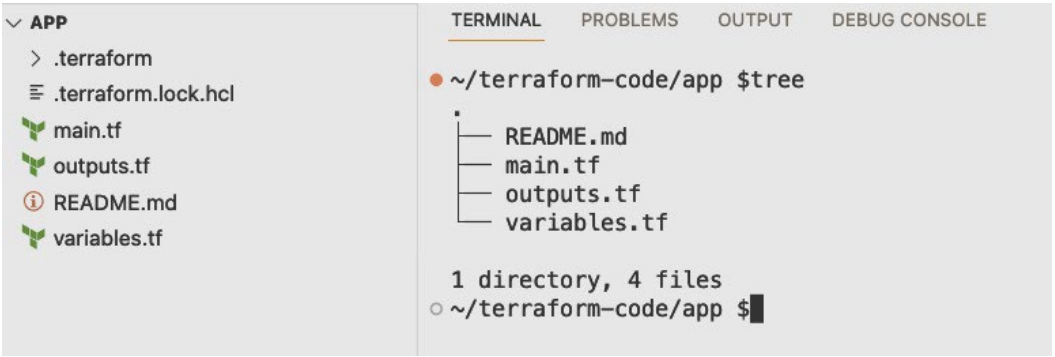

Figure 5.10: Module structure with minimal files

```
### GENERIC SYNTAX OF MODULE ###
module "local-name-for-the-module" {
  source    = "Local/Remote location of the child module"
  version   = "Version number of the module to be downloaded from the above source"
  VARIABLE1 = Value1
  VARIABLE2  = Value2
}

### EXAMPLE OF A MODULE ###
module "key_pair" {
  source  = "terraform-aws-modules/key-pair/aws"
  version = "2.0.2"

  key_name           = "my-keypair"
  create_private_key = true
}
```

Figure 5.11: Syntax and example of a module block

s

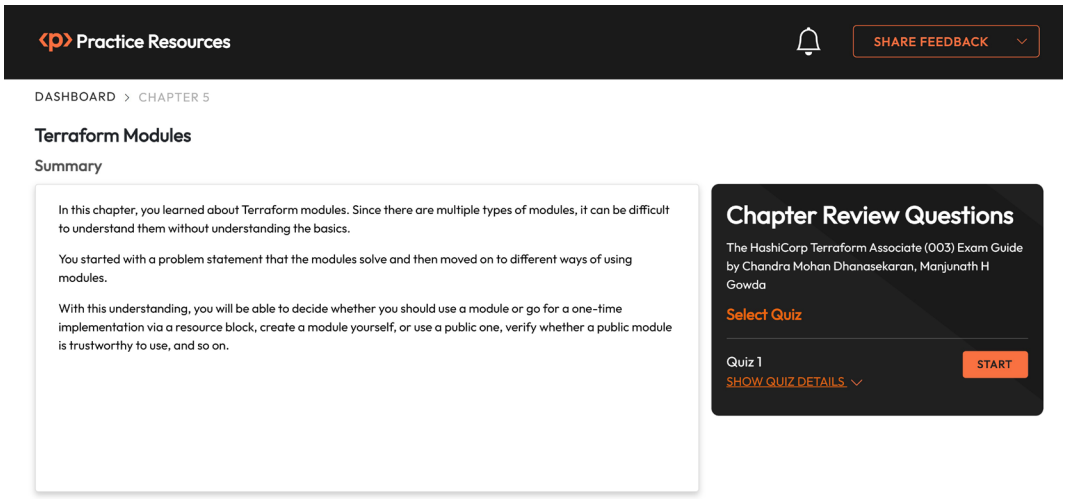

Figure 5.13 – Chapter Review Questions for Chapter 5

Chapter 6: Terraform Backends and Resource Management

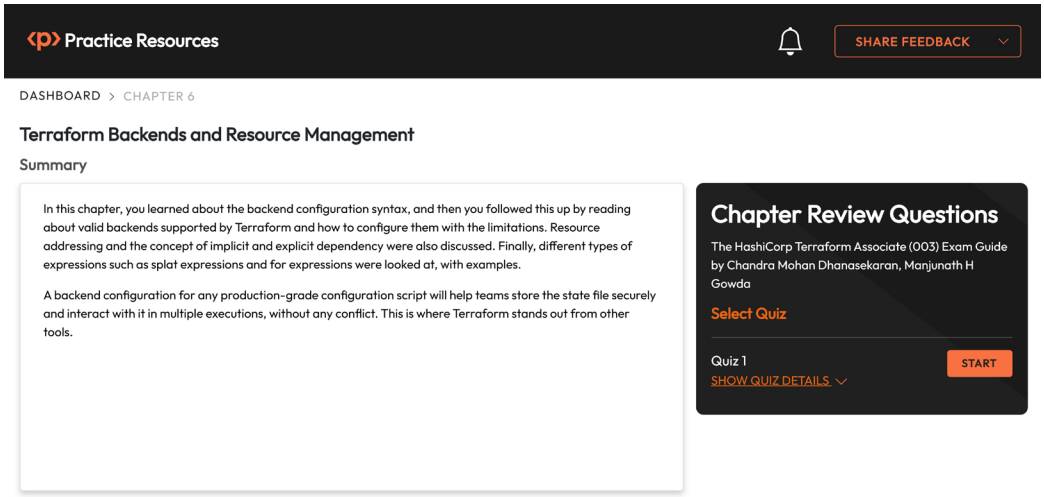

Figure 6.4 – Chapter Review Questions for Chapter 6

Chapter 7: Debugging and Troubleshooting Terraform

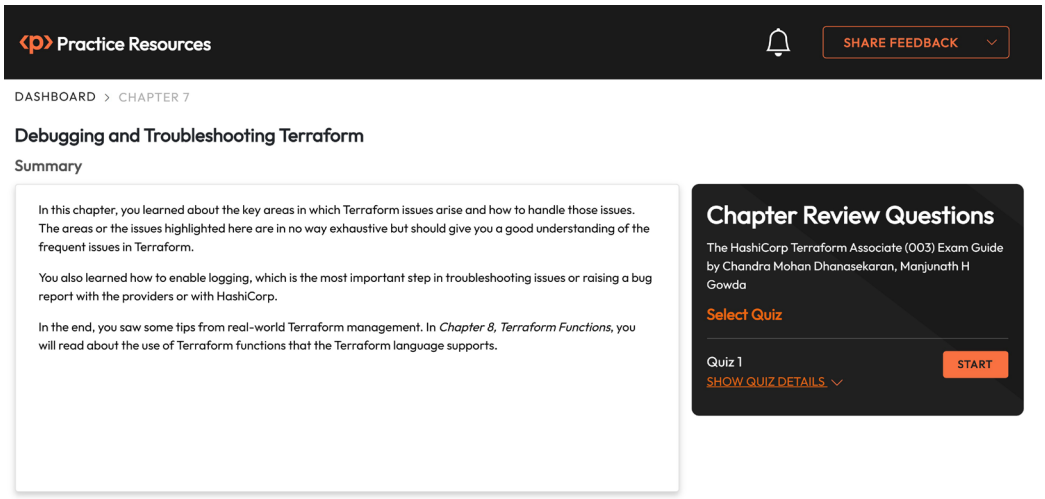

Figure 7.2 – Chapter Review Questions for Chapter 7

Chapter 8: Terraform Functions

main.tf > ...

```
variable "test-sensitive" {
  type      = string
  sensitive = true
  default   = "test-string-1"
}

variable "test-nonsensitive" {
  type      = string
  sensitive = false
  default   = "test-string-2"
}

output "outval1" {
  value     = var.test-sensitive
  sensitive = true
}

output "outval2" {
  value = var.test-nonsensitive
}

```

Figure 8.1 – main.tf with sensitive and nonsensitive variables

```
rm-functions-new\type-conversion-functions> terraform plan

Changes to Outputs:
  + outval1 = (sensitive value)
  + outval2 = "test-string-2"

You can apply this plan to save these new output values to the Terraform state, without changing any real infrastructure.
```

Figure 8.2 – The terraform plan terminal output

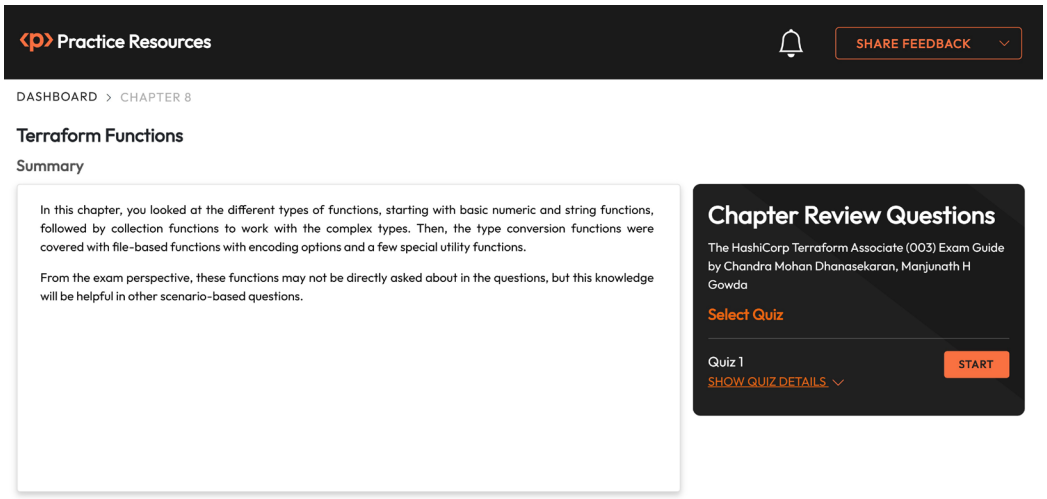

Figure 8.4 – Chapter Review Questions for Chapter 8

Chapter 9: Understanding HCP Terraform's Capabilities

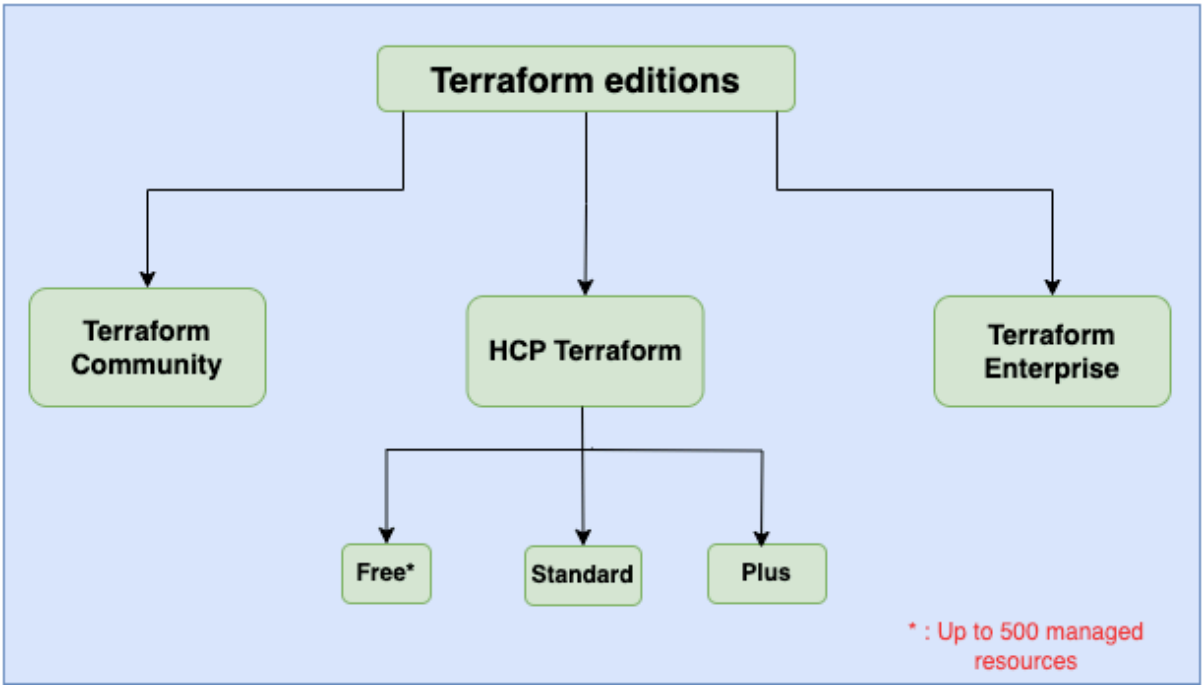

Figure 9.1: Terraform editions

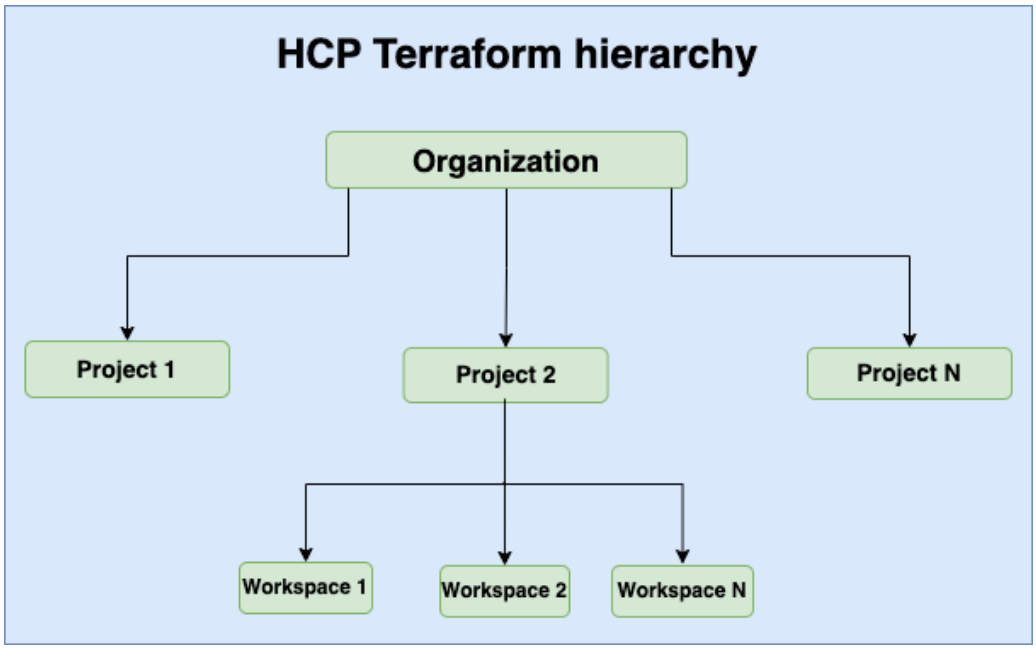

Figure 9.2: HCP Terraform concepts hierarchy

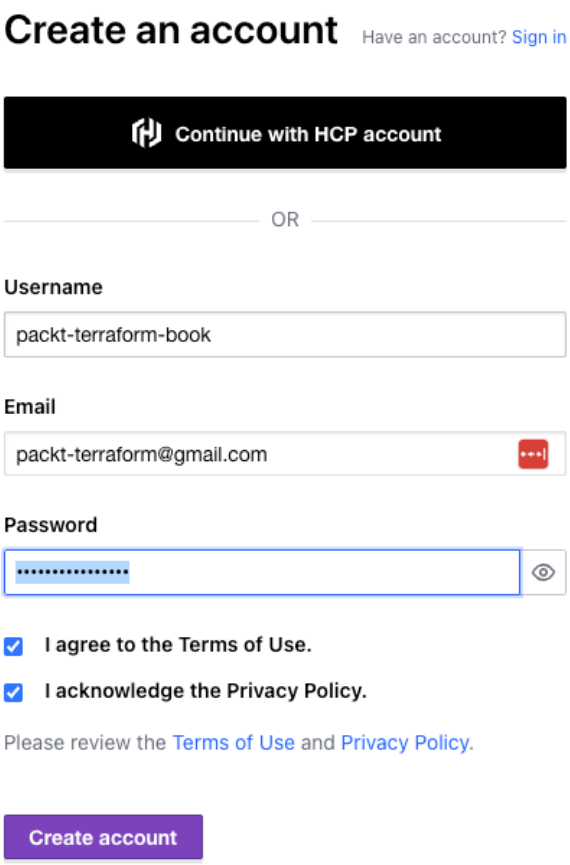

Figure 9.3: HCP Terraform account sign up

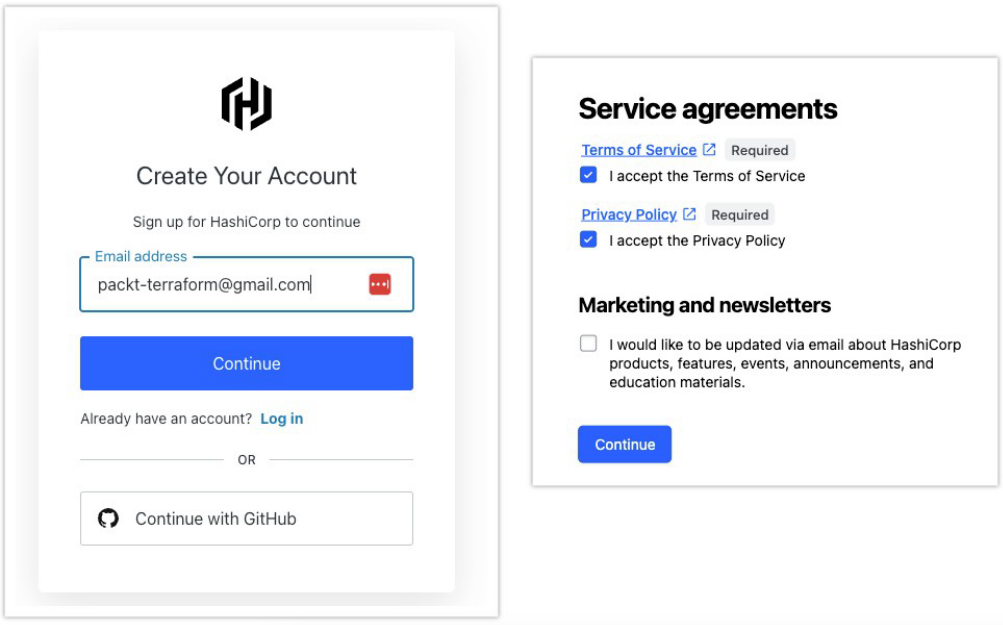

Figure 9.4: HCP sign-up screens

```
~ $terraform login
Terraform will request an API token for app.terraform.io using your browser.

If login is successful, Terraform will store the token in plain text in
the following file for use by subsequent commands:
    /Users/packt/.terraform.d/credentials.tfrc.json

Do you want to proceed?
  Only 'yes' will be accepted to confirm.

  Enter a value:
```

Figure 9.5: HCP Terraform login via CLI

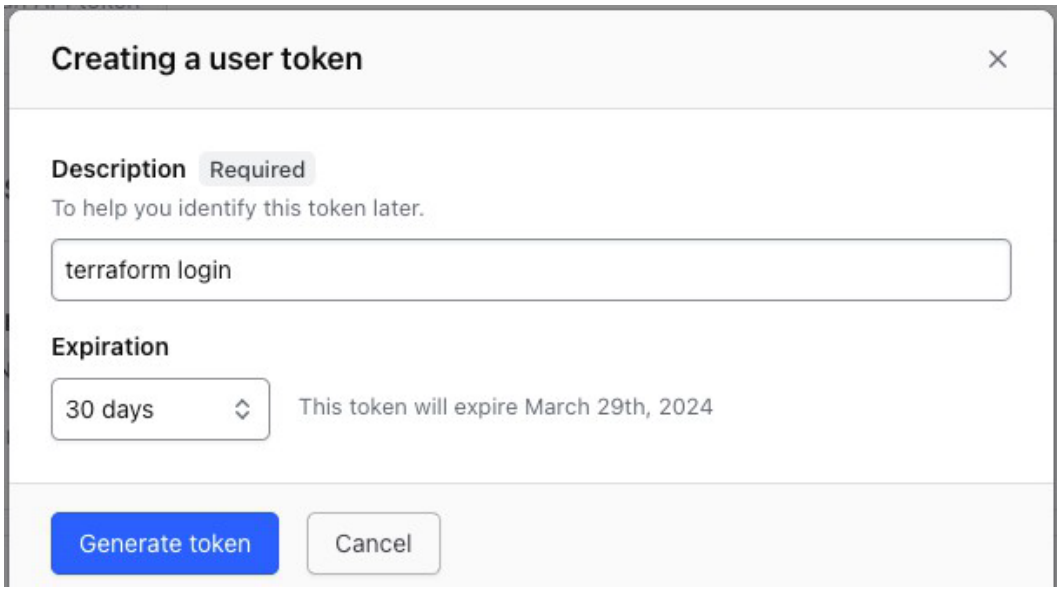

Figure 9.6: API token creation in HCP Terraform

```
-------------------------------------------------------------------------------
-

Terraform must now open a web browser to the tokens page for app.terraform.io.

If a browser does not open this automatically, open the following URL to proceed
:
    https://app.terraform.io/app/settings/tokens?source=terraform-login


-------------------------------------------------------------------------------
-

Generate a token using your browser, and copy-paste it into this prompt.

Terraform will store the token in plain text in the following file
for use by subsequent commands:
    /Users/packt/.terraform.d/credentials.tfrc.json

Token for app.terraform.io:
  Enter a value:
```

Figure 9.7: HCP Terraform login via CLI waiting for API token

1.

```
Welcome to HCP Terraform!

Documentation: terraform.io/docs/cloud

New to HCP Terraform? Follow these steps to instantly apply an example configuration:

  $ git clone https://github.com/hashicorp/tfc-getting-started.git
  $ cd tfc-getting-started
  $ scripts/setup.sh
```

Figure 9.8: Successful login to HCP Terraform via the CLI

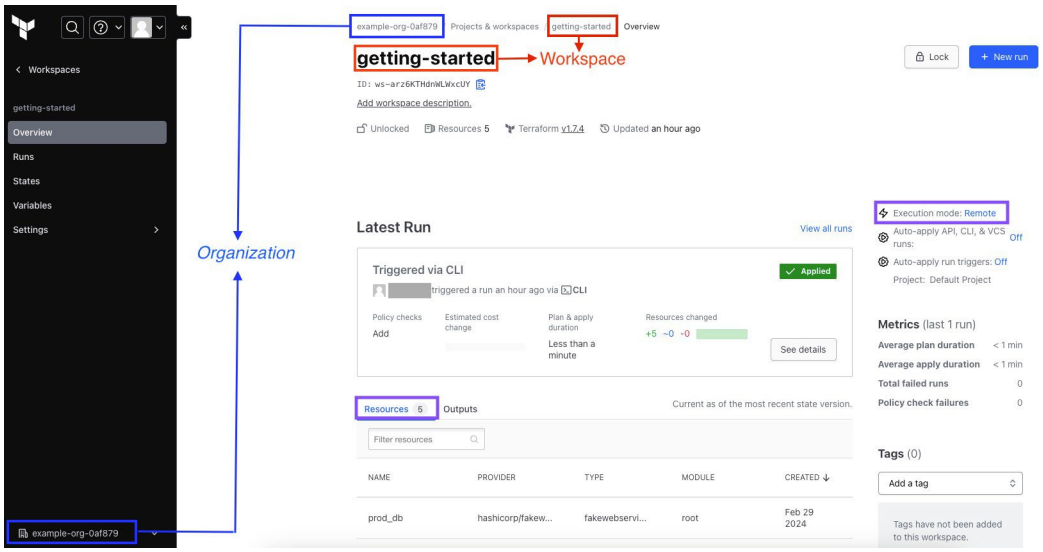

Figure 9.9: CLI-driven remote execution mode completion details

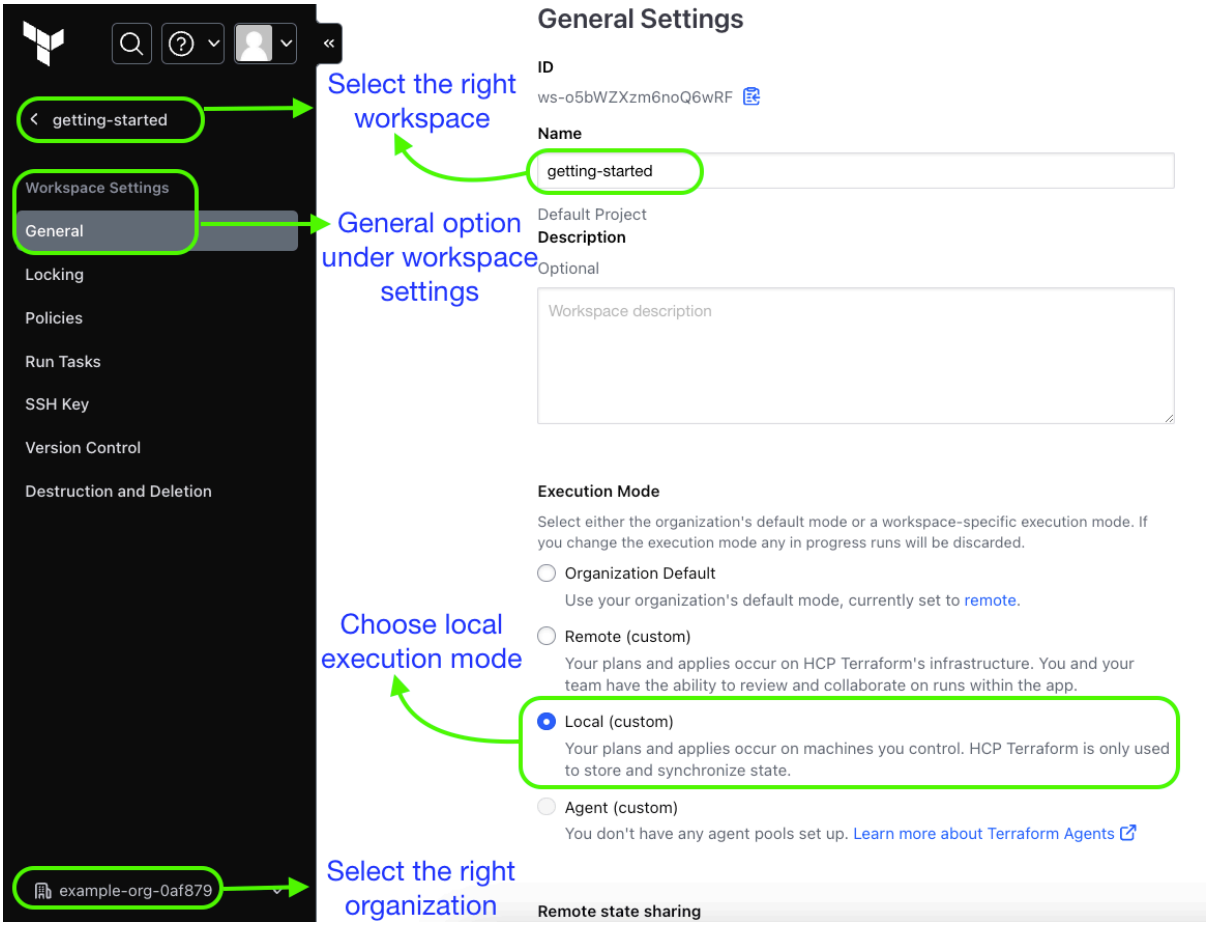

Figure 9.10: CLI-driven local execution mode configuration

```
~/tfc-getting-started $ls
LICENSE         README.md       backend.tf      main.tf         provider.tf     scripts
~/tfc-getting-started $cat main.tf
resource "fakewebservices_vpc" "primary_vpc" {
  name       = "Primary VPC"
  cidr_block = "10.0.0.0/8"
}

resource "fakewebservices_server" "servers" {
  count = 10

  name = "Server ${count.index + 1}"
  type = "t2.micro"
  vpc  = fakewebservices_vpc.primary_vpc.name
}

resource "fakewebservices_load_balancer" "primary_lb" {
  name    = "Primary Load Balancer"
  servers = fakewebservices_server.servers[*].name
}

resource "fakewebservices_database" "prod_db" {
  name = "Production DB"
  size = 256
}
```

cidr_block is changed to 10.0.0.0/8

Number of servers changed to 10

Figure 9.11: CLI-driven local execution mode changes

```
Plan: 8 to add, 2 to change, 0 to destroy.
```

Figure 9.12: `terraform plan` output for local execution mode

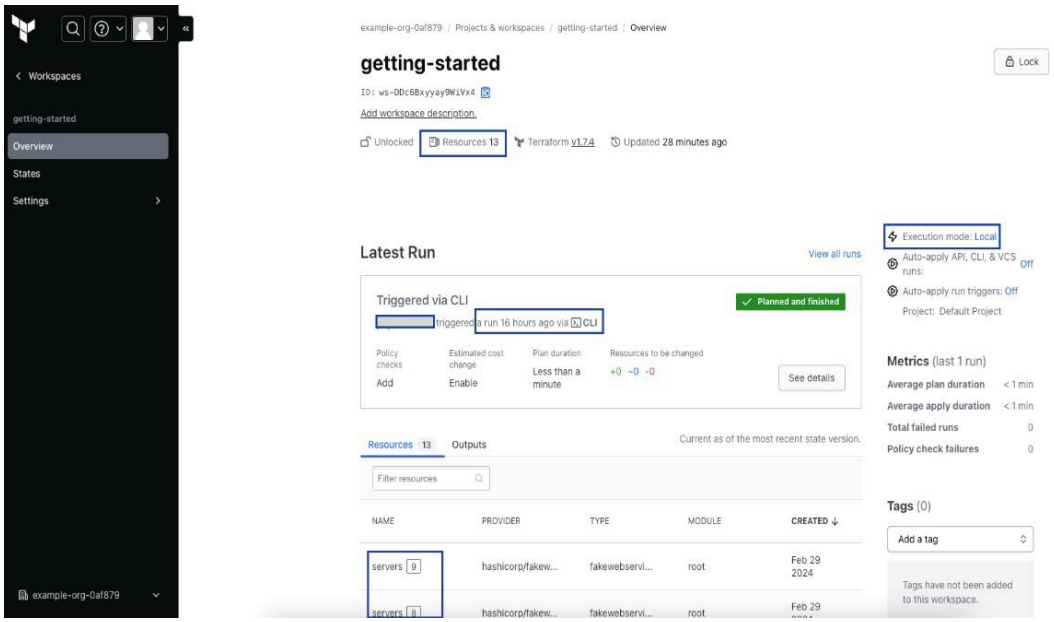

Figure 9.13: CLI-driven local execution mode completion details

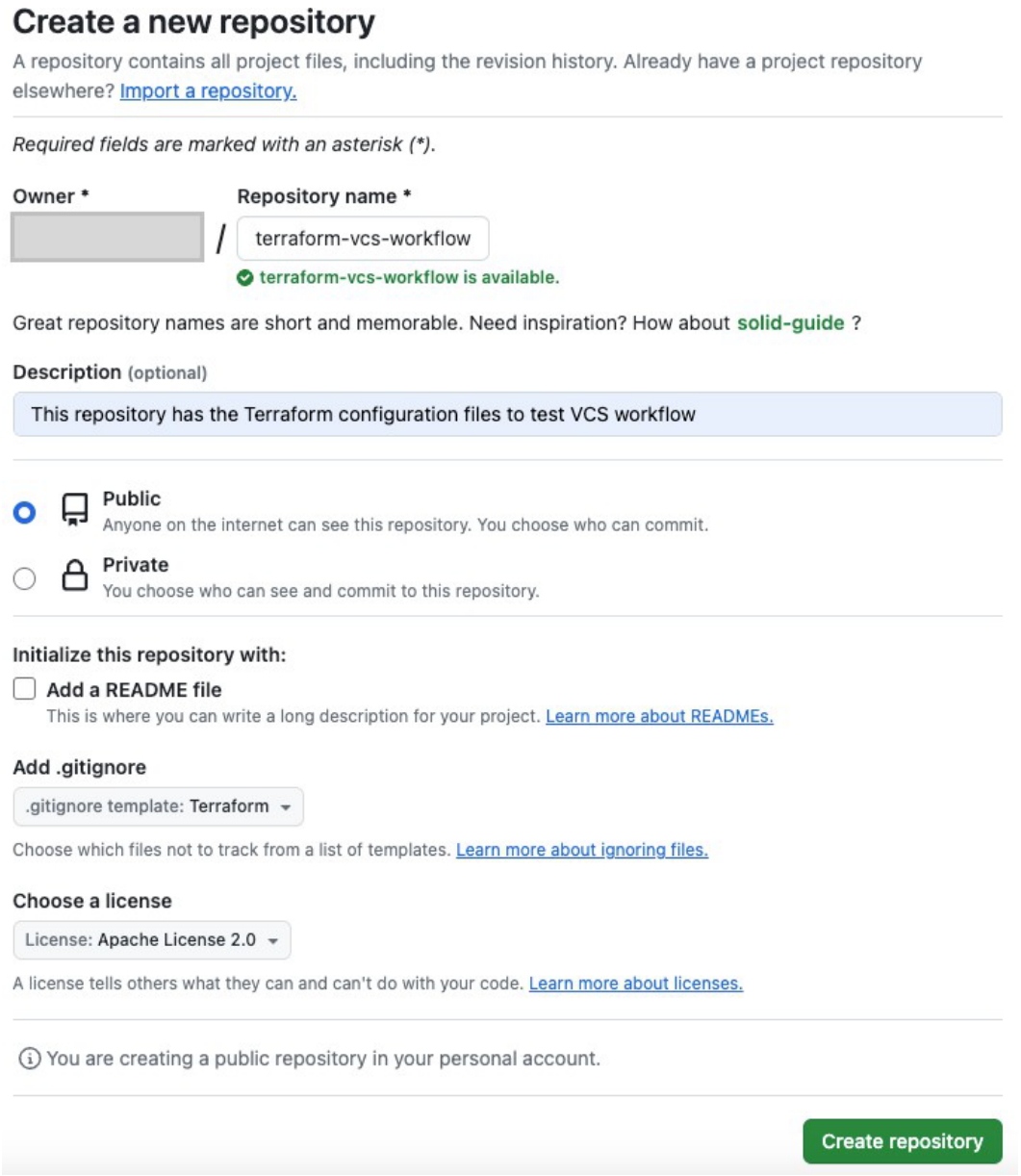

Figure 9.14: Creating a public repository in GitHub

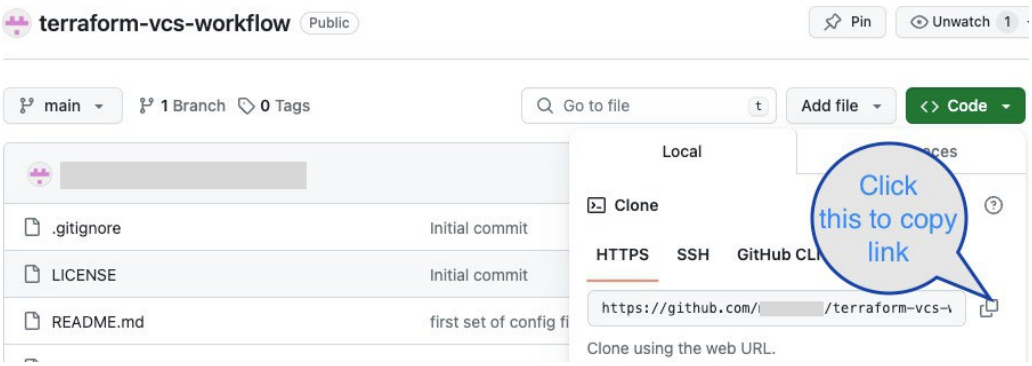

Figure 9.15: Copying the HTTPS link from the GitHub repository

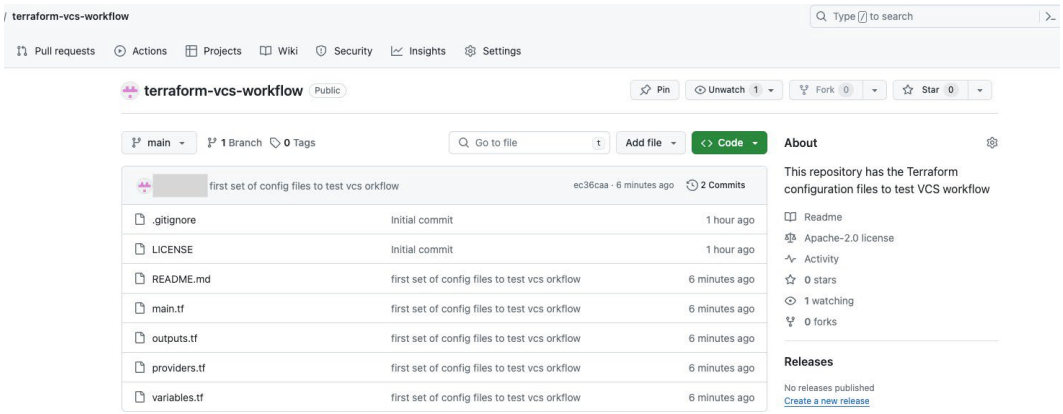

Figure 9.16: Exercise files populated in the GitHub repository

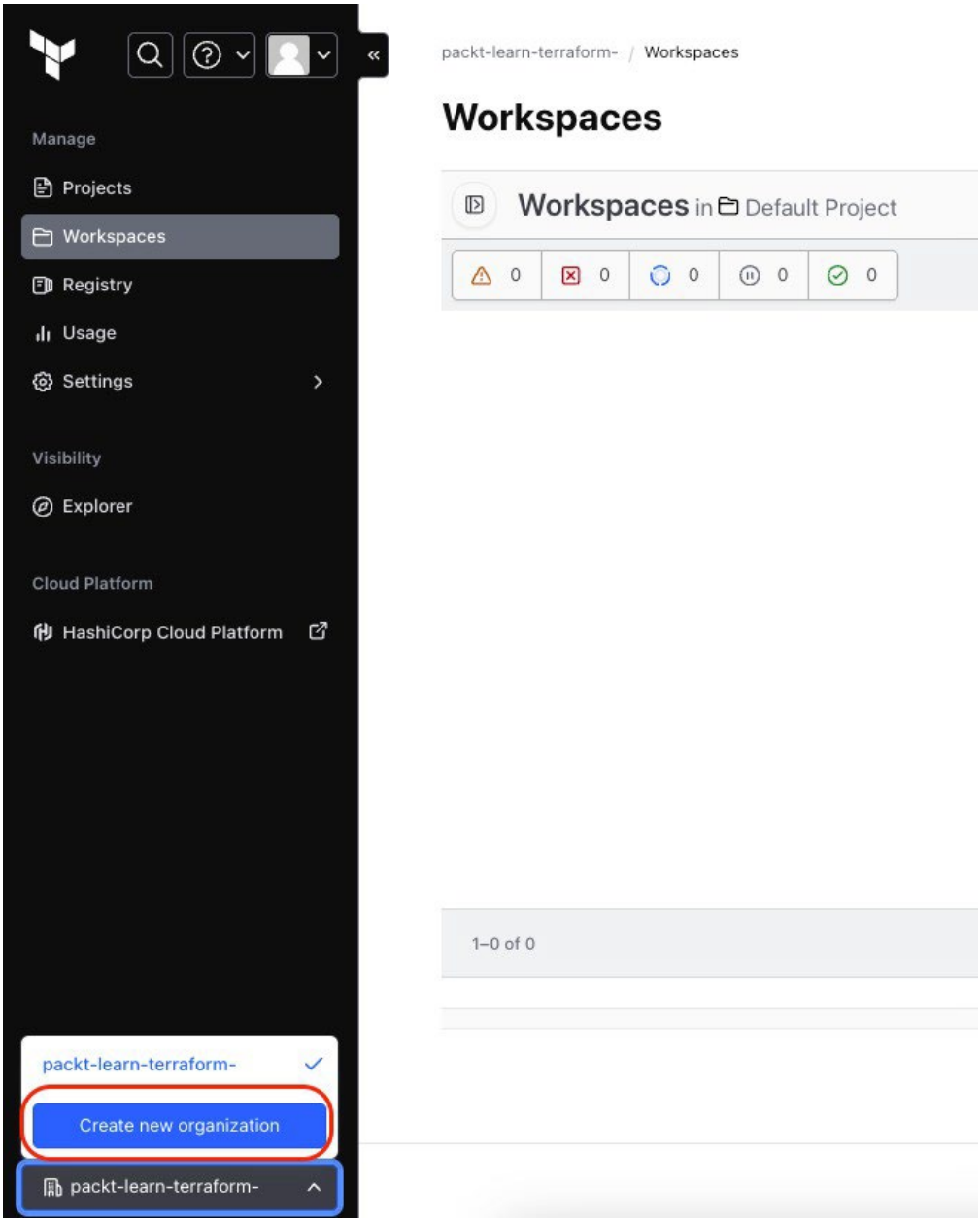

Figure 9.17: Creating a new organization in HCP Terraform

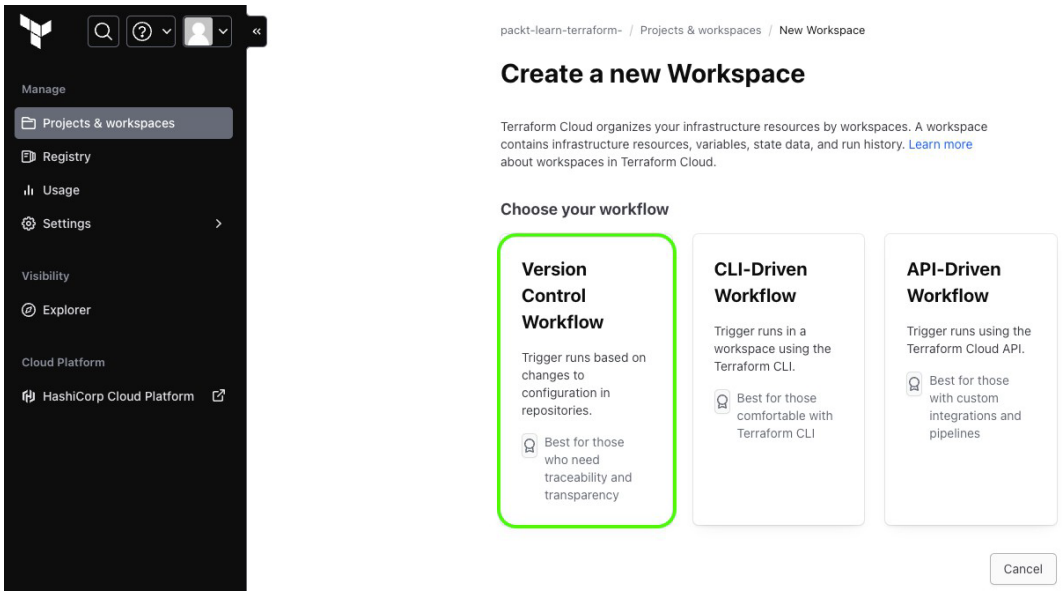

Figure 9.18: Creating a new workspace with `Version Control Workflow`

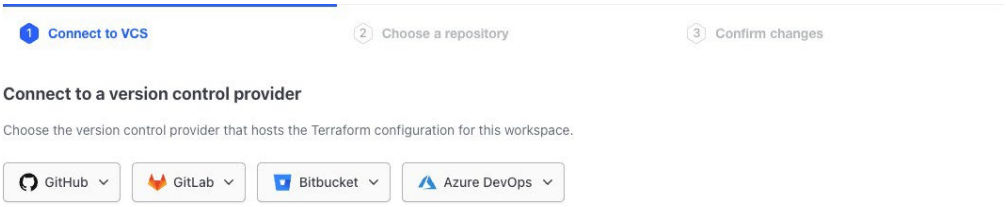

Figure 9.19: Choosing the version control provider

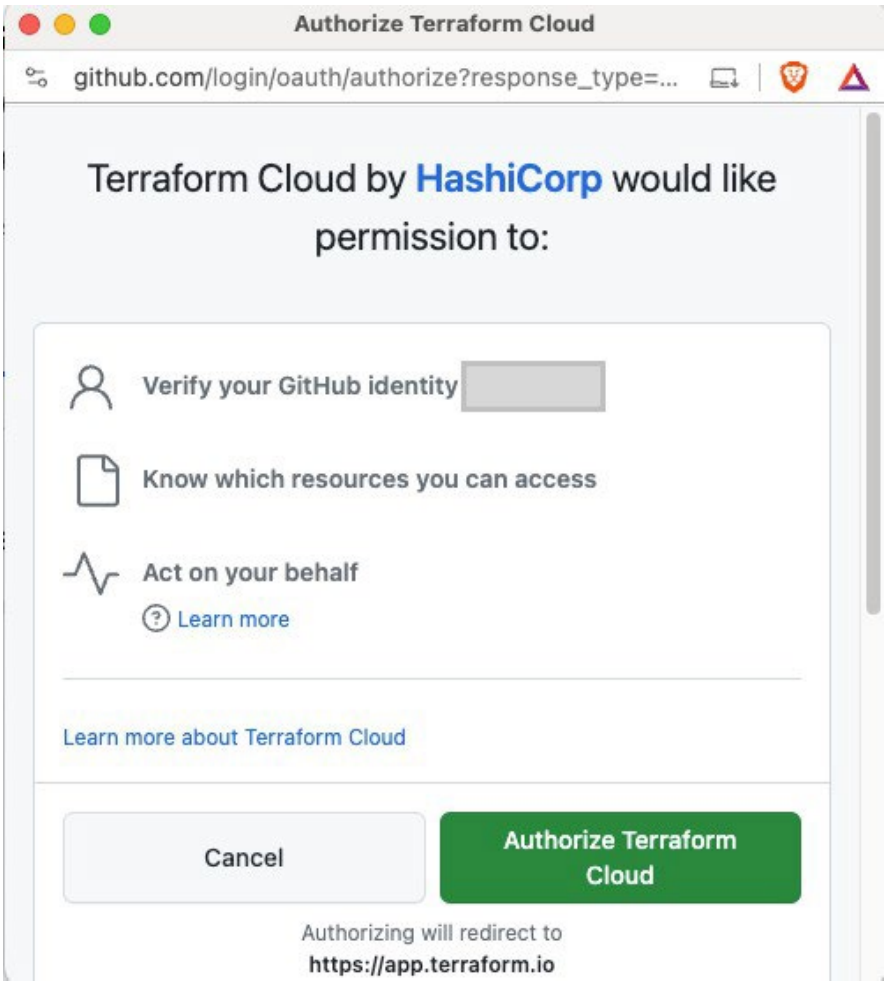

Figure 9.20: Authorization window in GitHub

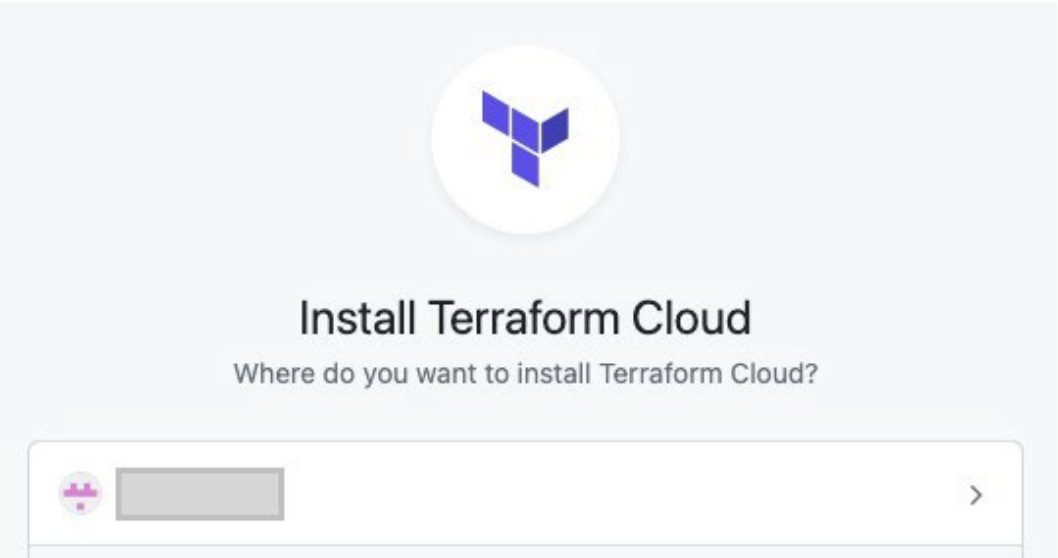

Figure 9.21: Install Terraform Cloud/HCP Terraform in GitHub

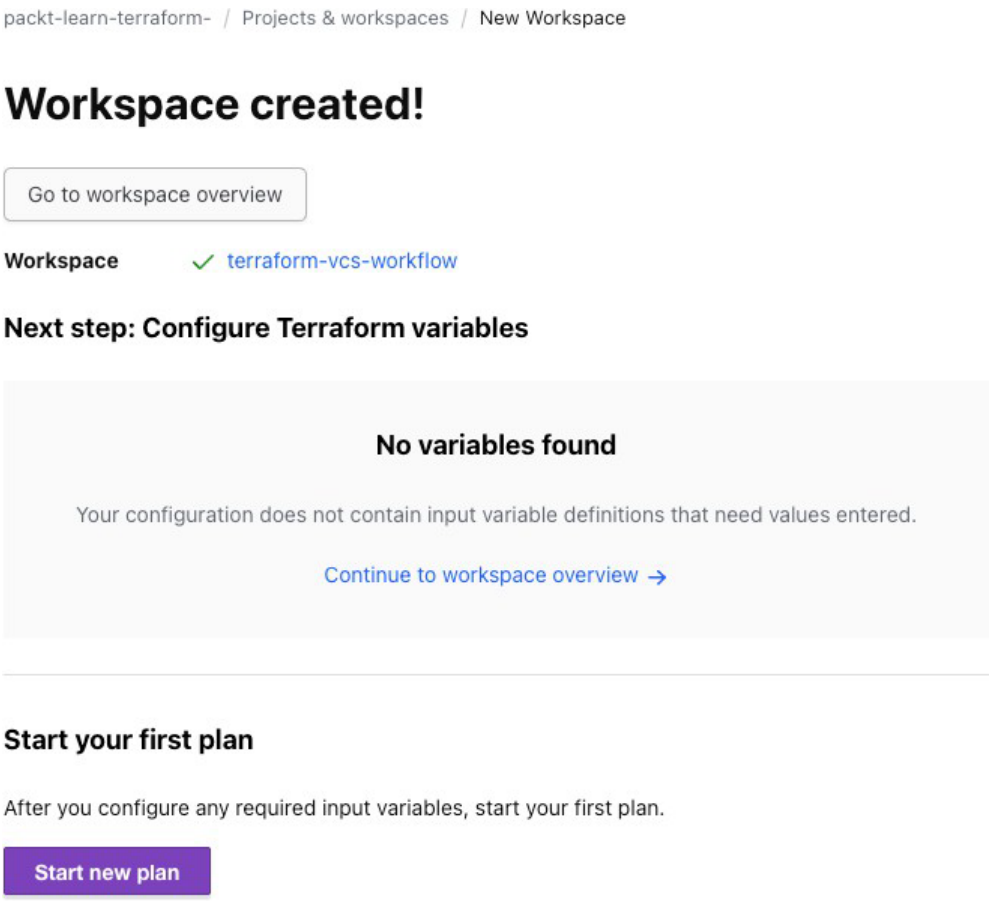

Figure 9.22: Successful creation of a workspace

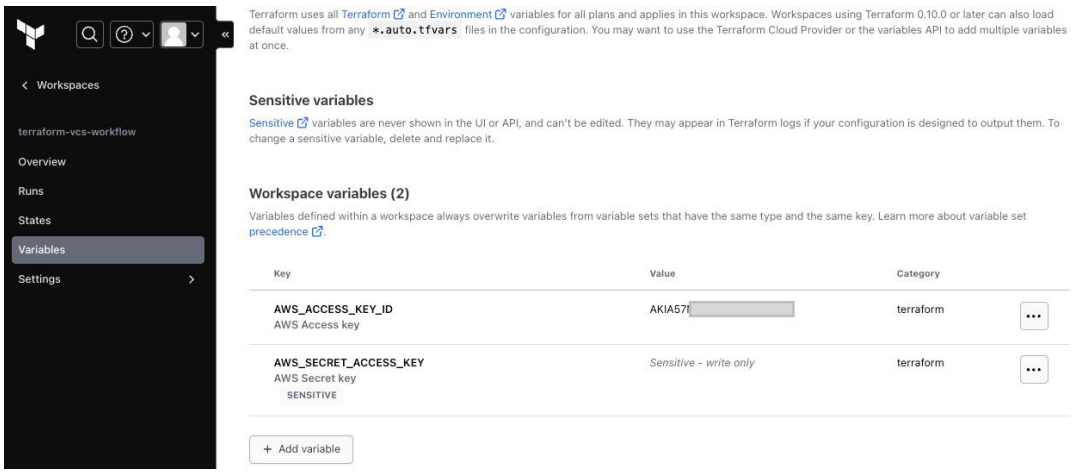

Figure 9.23: Adding AWS credentials as variables

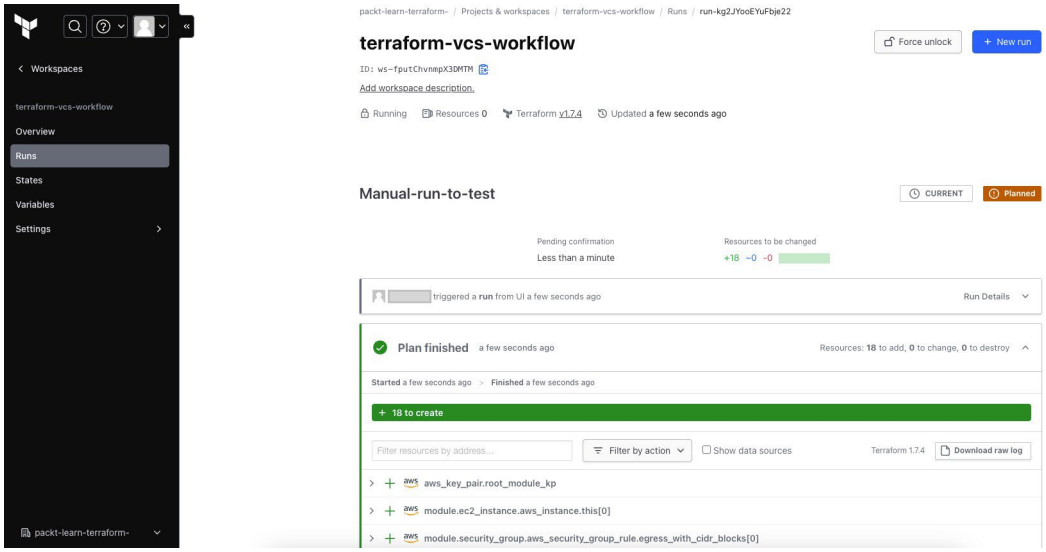

Figure 9.24: terraform plan output for VCS trigger

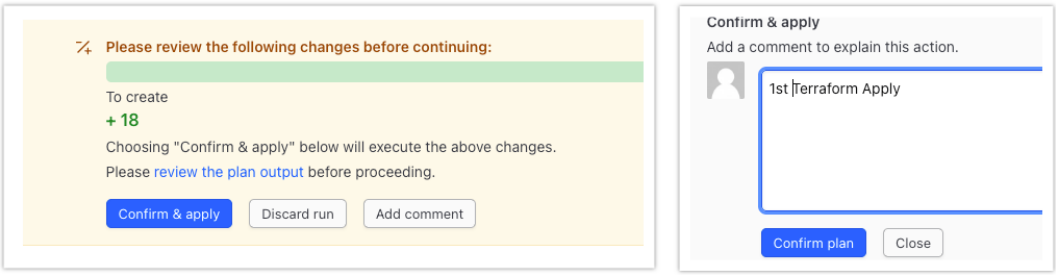

Figure 9.25: Confirm the `Terraform plan` to apply

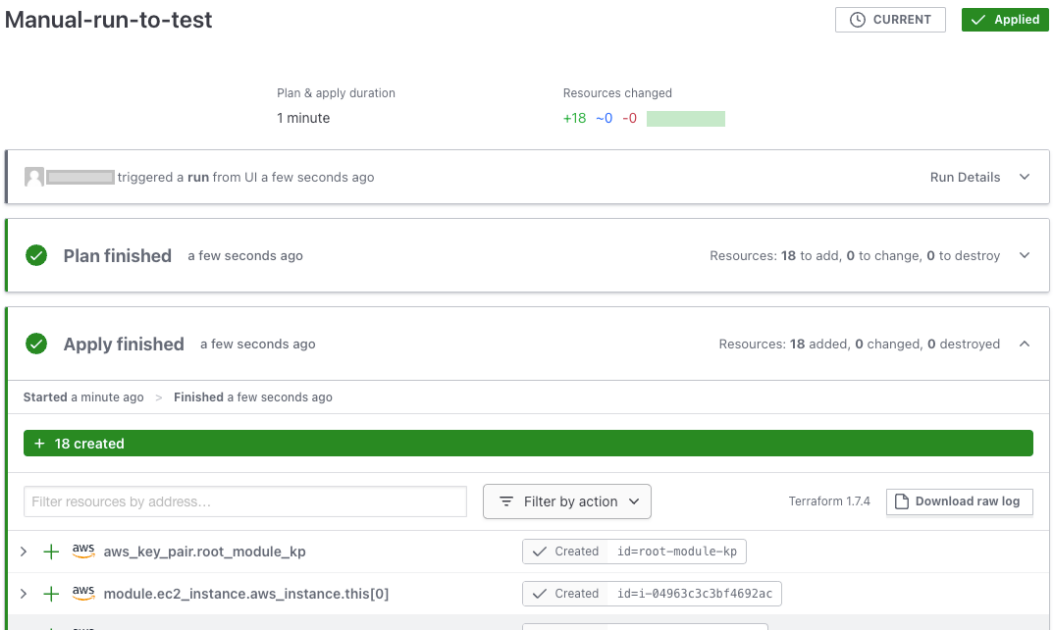

Figure 9.26: Creation of resources after a manual run

```
resource "aws_key_pair" "root-module-kp" {
  key_name   = "vcs-test-kp"
```

Figure 9.27: Changing the keypair name

-

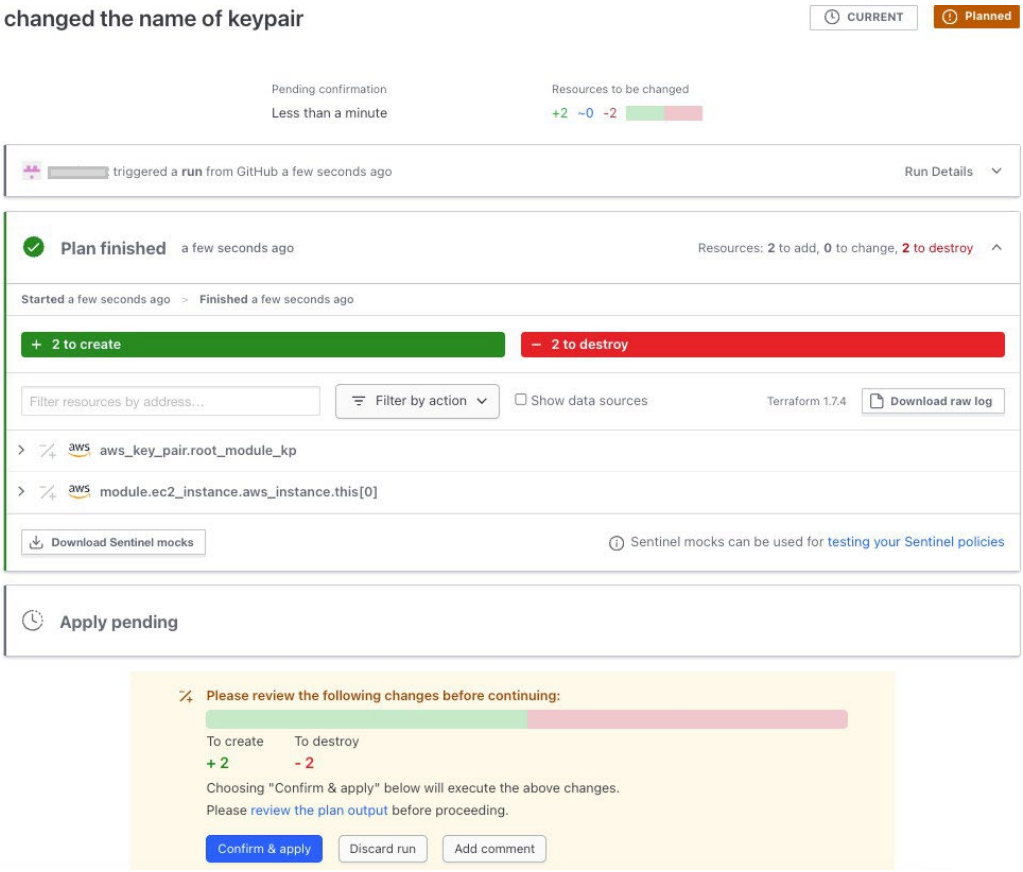

Figure 9.28: terraform plan output for the keypair change in the VCS repository

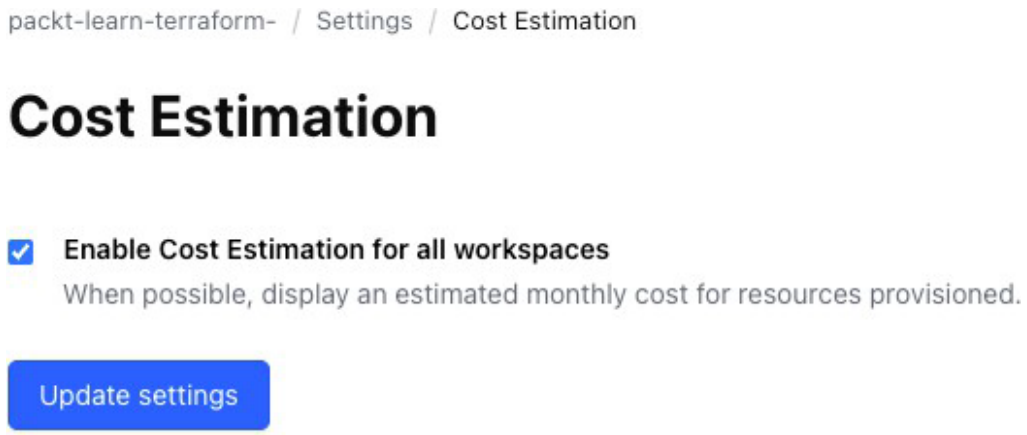

Figure 9.29: Enabling the Cost Estimation feature at the organization level

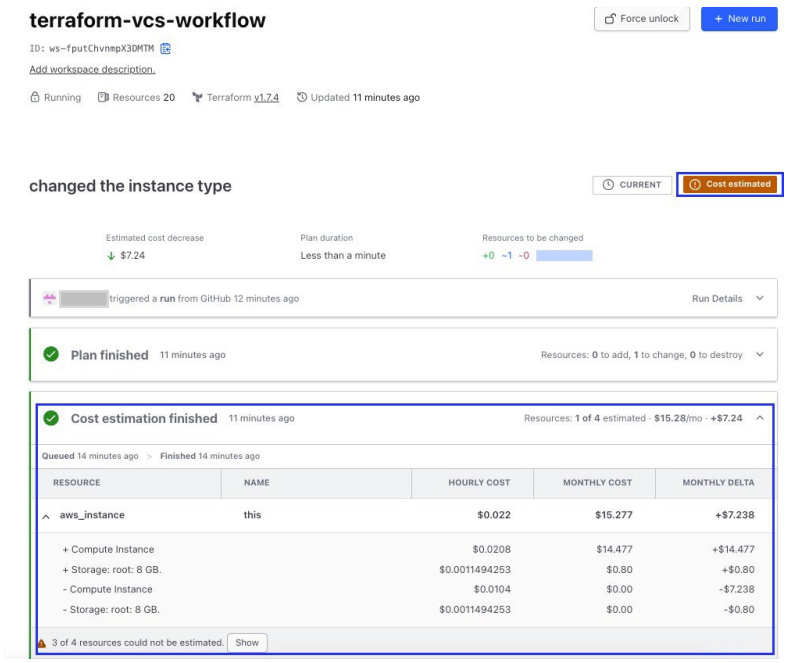

Figure 9.30: terraform plan showing the estimated cost

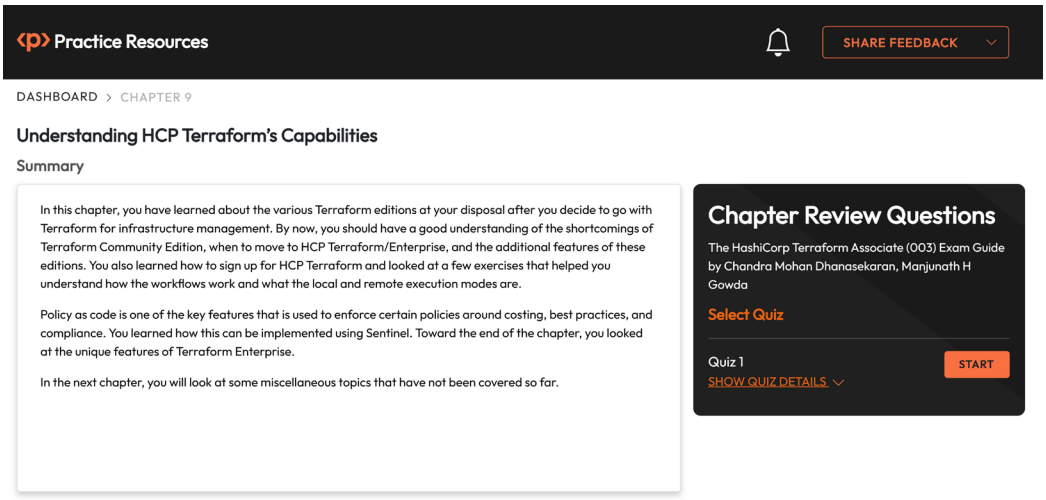

Figure 9.32 – Chapter Review Questions for Chapter 9

```
Error: Invalid value for variable

  on input-validations.tf line 5:
   5: variable "app_id" {
      ├────────────────
      │ var.app_id is "app-1234"

The app_id value must include the prefix "app-" and should be 9 characters long.

This was checked by the validation rule at input-validations.tf:9,3-13.
```

Figure 10.1 – Error message returned by Terraform

```
Plan: 1 to add, 0 to change, 0 to destroy.

  Warning: Check block assertion failed

    on check-assertions.tf line 7, in check "destroy_check":
     7:       condition = aws_s3_bucket.app_bucket.force_destroy == true
        ├────────────────
        │ aws_s3_bucket.app_bucket.force_destroy is false

  The S3 bucket created with the force_destroy parameter as false
```

Figure 10.2 – Error message from the check {} block

```
aws_s3_bucket.app_bucket: Creating...
aws_s3_bucket.app_bucket: Creation complete after 6s [id=terraform-

  Warning: Check block assertion failed

    on check-assertions.tf line 7, in check "destroy_check":
     7:       condition = aws_s3_bucket.app_bucket.force_destroy == true
        ├────────────────
        │ aws_s3_bucket.app_bucket.force_destroy is false

  The S3 bucket created with the force_destroy parameter as false


Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

Figure 10.3 – Output from terraform apply

```
aces-example> terraform workspace list
  default
* dev
```

Figure 10.4 – terraform workspace list command output

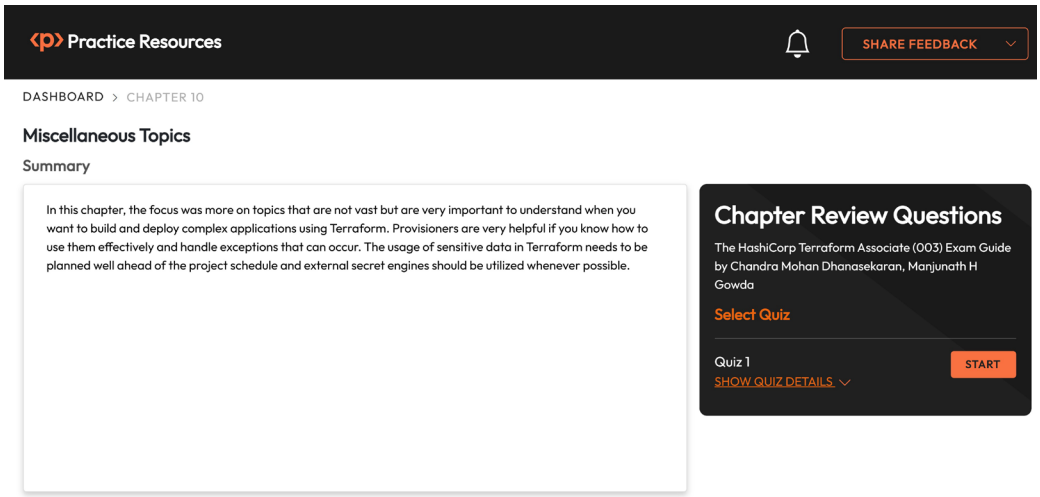

Figure 10.6 – Chapter Review Questions for Chapter 10

Chapter 11: Accessing the Online Practice Resources

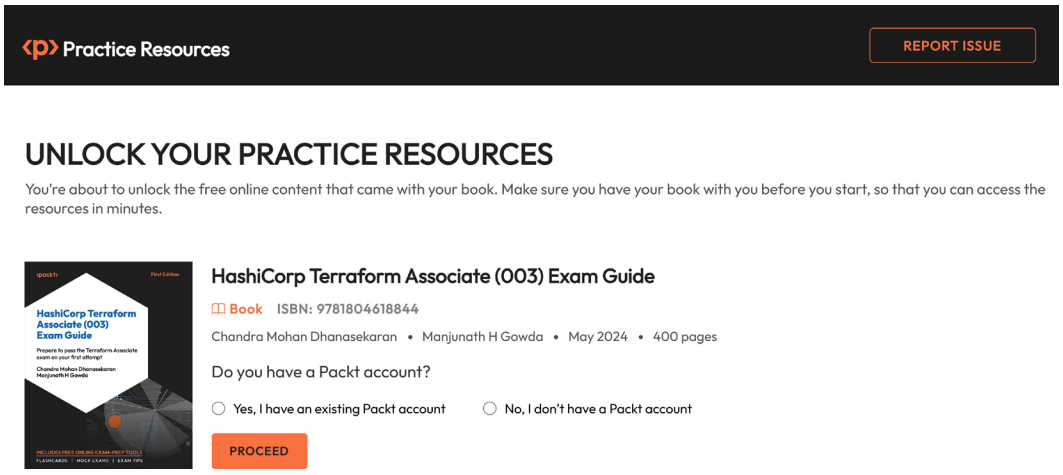

Figure 11.2 – Unlock page for the online practice resources

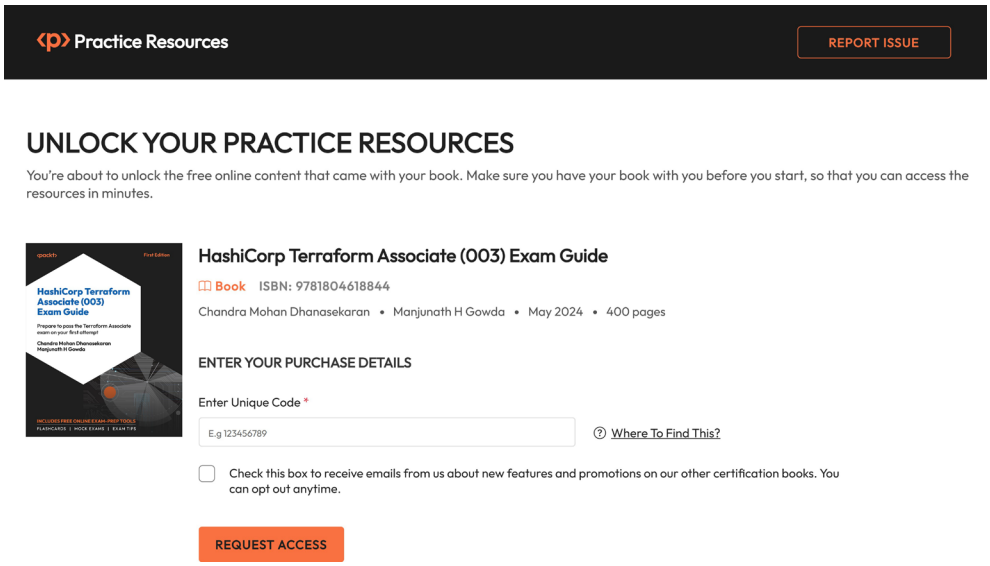

Figure 11.3 – Enter your unique sign-up code to unlock the resources

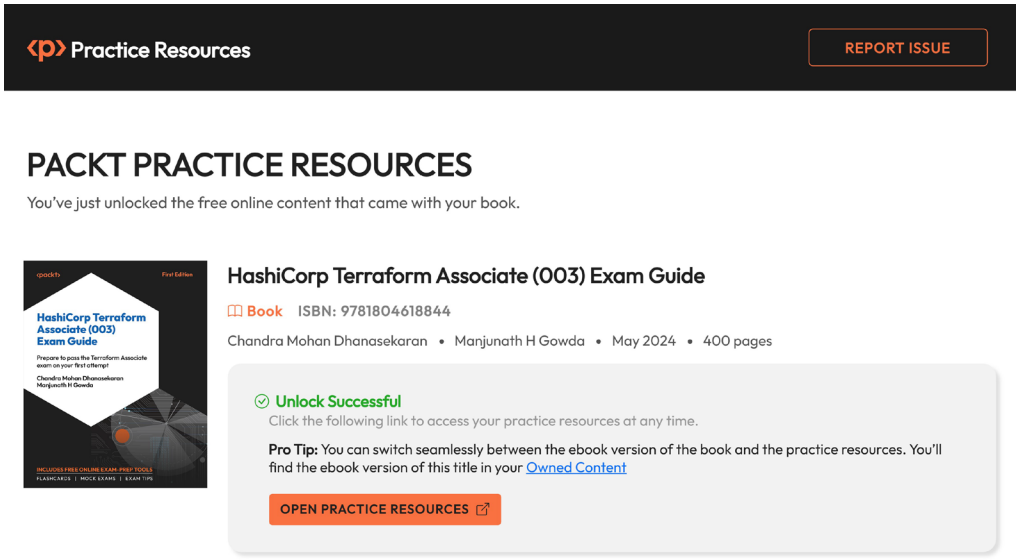

Figure 11.4 – Page that shows up after a successful unlock

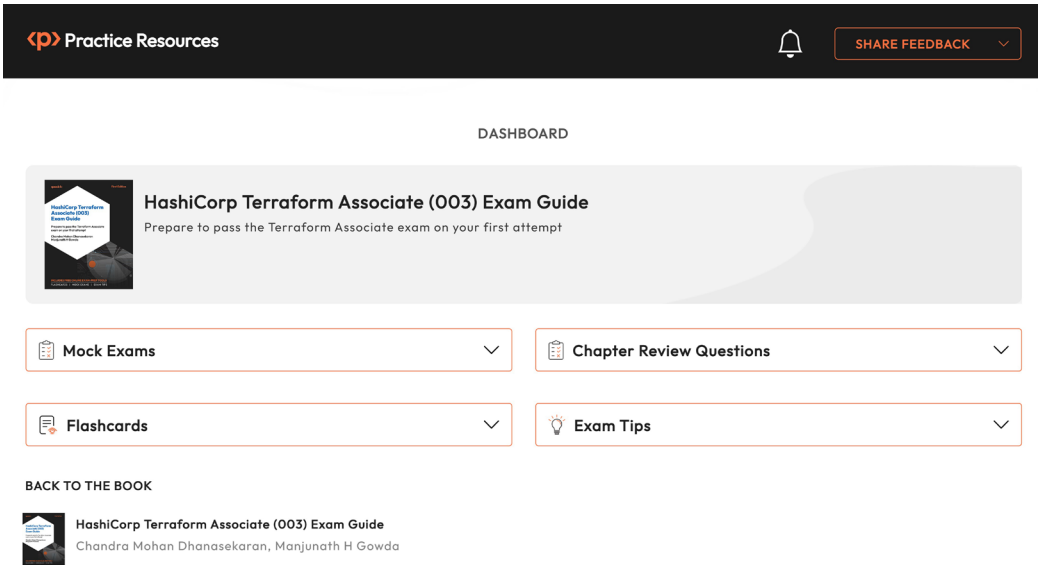

Figure 11.5 – Dashboard page for HashiCorp Terraform Associate (003) practice resources

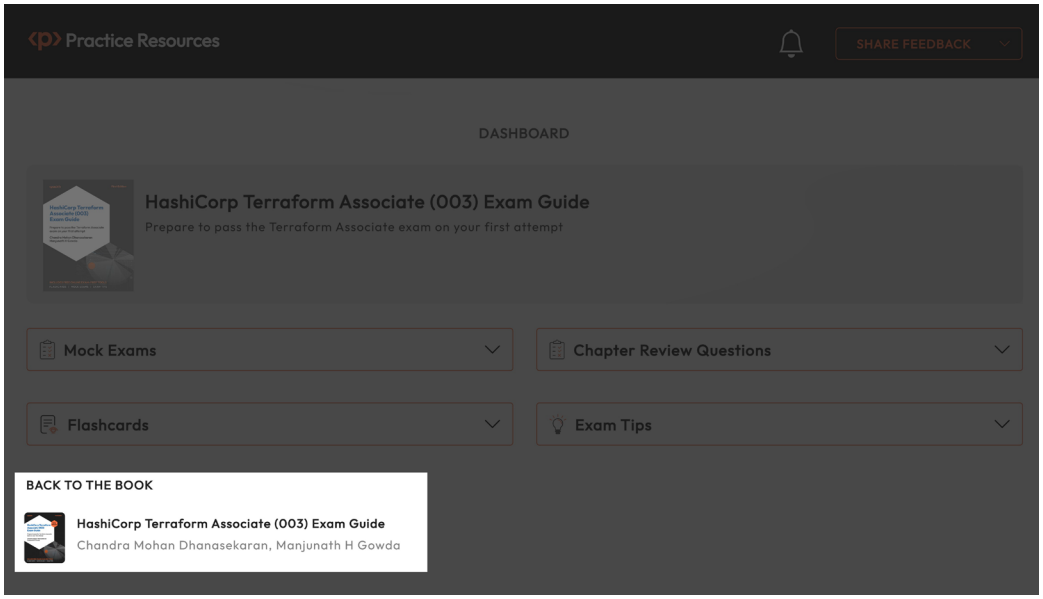

Figure 11.7 – Dashboard page for HashiCorp Terraform Associate (003) practice resources